MARCO COPPELLI
BRUNO STORTONI

MICROPROCESSORI

TEORIA E PROGETTO

2ª EDIZIONE
AMPLIATA

LA SOVRANA EDITRICE
FERMO

Edizioni della «Sovrana»:

Esercizi di Elettronica generale - Vol. I.

Esercizi di Elettronica generale - Vol. II.

Tecnologia delle Costruzioni elettroniche - Vol. I.

Tecnologia delle Costruzioni elettroniche - Vol. II.

Tecnologia delle Costruzioni elettroniche - Vol. III.

Gestione di archivi e linguaggio COBOL

Conoscere l'Informatica

Elettronica Digitale

Esercizi di Elettronica Digitale

Microprocessori - Teoria e Progetto

Elettronica Generale - Dispositivi

Elettronica Generale - Integrati

Radioelettronica

Sistemi - Automatismi - Informazione

Esercizi di Elettrotecnica - Vol. I.

Esercizi di Elettrotecnica - Vol. II.

Le Fibre Ottiche nelle Telecomunicazioni

Telegrafia - Telefonia - Ponti Radio

Trasmissioni Numeriche e
Sistemi su Fibra Ottica

MARCO COPPELLI - BRUNO STORTONI

MICROPROCESSORI TEORIA E PROGETTO

2ª EDIZIONE
AMPLIATA

LA SOVRANA EDITRICE - FERMO
63023 Fermo - Via Alfieri 12 - Tel. (0734) 32257

Collana di testi scientifici per gli allievi degli Istituti Tecnici Industriali e Professionali diretta dall'Ing. Giuseppe Lotti, titolare della Cattedra di Elettronica presso l'I.T.I.S. «Montani» di Fermo.

© Stampato in Italia
Marzo 1986

PROPRIETÁ LETTERARIA, ARTISTICA, SCIENTIFICA RISERVATA

stampa Ind. Grafiche Fratelli Aniballi
Zona Ind. Baraccola - Ancona

disegni Gioacchino Fasino

copertina Clara Romagnoli

PREFAZIONE

La tecnologia elettronica è entrata in un'era di crescita esponenziale, ed i cambiamenti di ordine economico e sociale ad essa connessi interverranno nella sua scia con questo stesso ritmo grazie al microprocessore.

Il testo, che fa seguito al volume «Elettronica Digitale», ne mantiene l'impostazione prevalentemente sperimentale ed è frutto dell'esperienza maturata dagli autori nei corsi tenuti anche presso aziende ed enti diversi.

La sua peculiare caratteristica è di esporre i circuiti e le applicazioni in forma completa ed interamente definita. Lo scopo è infatti, non solo quello di introdurre alla conoscenza di questa cellula base del nuovo universo dell'informatica, ma anche quello di permettere la progettazione e realizzazione di sistemi programmabili a complessità crescente man mano che si procede nello studio.

Il testo si rivolge a quanti, studenti e progettisti, in possesso sia pur delle nozioni fondamentali dell'elettronica digitale, vogliano utilizzare il microprocessore nelle sue molteplici applicazioni.

Gli autori desiderano ringraziare le case costruttrici di circuiti integrati e strumentazione, la Alberti Elettronica ed in particolare la SGS che ha permesso di pubblicare il set di istruzioni dello Z 80, al quale viene fatto particolare riferimento nel testo. Sono inoltre grati al P.I. Gioacchino Fasino che ha eseguito perfettamente i disegni contenuti nel volume, ed alla casa editrice «La Sovrana» che ne ha permesso la pubblicazione.

Fermo, Dicembre 1982 Gli Autori

PREFAZIONE ALLA SECONDA EDIZIONE

La rilevante variazione di questa edizione, rispetto alla precedente, è dovuta all'aggiunta del capitolo decimo che è indirizzato soprattutto ai lettori più esperti nel campo dei microprocessori.

In esso vengono esposte l'analisi e la sintesi relativa a: test per circuiti integrati; sistema minimo a microprocessore; semplice videogioco.

Si vuole, così, venire incontro alle esigenze di operatori che hanno espressamente richiesto applicazioni che prescindano dall'uso di un particolare microcomputer.

Nella speranza di aver fatto cosa gradita si esprime il ringraziamento a quanti hanno fatto pervenire suggerimenti per il miglioramento del testo.

Fermo, Marzo 1986 Gli Autori

INDICE GENERALE

CAPITOLO PRIMO **ELETTRONICA PROGRAMMABILE**

1. **Introduzione** 1
2. **Uso di una memoria PROM come circuito combinatorio**
   Reti logiche programmabili 2
3. **Uso di una PROM come circuito sequenziale** 8
   a) disegno del diagramma temporale delle uscite 10
   b) tabella di flusso degli stati di uscita 12
   c) tabella delle eccitazioni di uscita 13
   d) tabella di programmazione 14
4. **Struttura di un microcomputer** 15
5. **L'hardware del microprocessore** 20
   Registro delle istruzioni e logica di controllo di microprogramma 21
   Unità aritmetico-logica (ALU) 22
   Registri della CPU 22
6. **Segnali della CPU** 24
   Esercizi proposti 30

CAPITOLO SECONDO **LA PROGRAMMAZIONE**

1. **Generalità** 31
   Analisi del problema 31
   Diagramma di flusso 31
   Codifica del programma 34
   Messa a punto del programma 34
2. **Linguaggi di programmazione** 34
3. **Flag** 37
   Flag di carry 38
   Flag di addizione-sottrazione 38
   Flag di parità e overflow 38
   Flag di half-carry 38
   Flag di zero 38
   Flag di segno 38
4. **Set di istruzioni del microprocessore Z 80** 38
   Utilizzo della carta di riferimento per la stesura di semplici programmi 51
   Caricamento del contenuto del registro B nel registro C 51
   Esecuzione della somma di due numeri esadecimali con risultato ≤FF 53
   Rappresentazione binaria complemento a due 54
   Loop (anello) 55
   Loop di delay (ritardo) 58
   Posizionamento ad 1 del bit di peso 2^2 di un byte 60
5. **Sottoprogramma o subroutine** 62
   Istruzioni CALL e RET 63
   Istruzioni di PUSH qq e POP qq 66
   Istruzioni LD r, (IX + d), NEG, AND s 67
   Istruzioni OUT (C), r ed RST p 68

6. **Metodi di indirizzamento dello Z 80** 70
Indirizzamento immediato 70
Indirizzamento indiretto tramite coppie di registri 71
Indirizzamento indicizzato 71
Indirizzamento tramite stack-pointer 71
Indirizzamento mediante istruzioni di salto relativo 71
7. **Esercitazioni di software** 72
Addizione di tre numeri 72
Livelli di subroutine 73
Riempimento di una area di memoria con un dato costante (FILL) 76
Ordinamento in forma crescente di 256 numeri 77
Conversione binario BCD 80
Esercizi proposti 83

CAPITOLO TERZO **L'HARDWARE DEL MICROPROCESSORE**

1. **Lo scambio di informazioni** 85
a) trasferimento punto a punto 85
b) trasferimento a diffusione 88
c) trasferimento a collettore 93
d) trasferimento a bus 94
2. **Reti sequenziali come controllori** 97
Controllori autonomi 97
Controllori con ingressi esterni 100
Controllori con ingressi di ritorno 102
3. **Esecuzione di un programma** 104
4. **Le temporizzazioni e le linee di controllo** 110
Lettura di codice operativo 112
Lettura/ scritture in memoria 113
Cicli di ingresso/uscita 114
Richiesta del bus 115
Richiesta di interruzione 116
5. **Reset** 117
6. **Caratteristiche elettriche** 120
7. **Tecnologie a confronto** 124
Esercizi proposti 125

CAPITOLO QUARTO **DISPOSITIVI DI INPUT-OUTPUT E CIRCUITI D'INTERFACCIA**

1. **Dispositivi d'ingresso-uscita (I/O)** 129
Collegamento dei dispositivi di input/output (I/O) 129
Tecnica di collegamento Memory/Mapped I/O 130
Tecnica di collegamento I/O isolato (I/O Mapped I/O) 133
2. **L'interrupt (Interruzione)** 136
Richiesta d'interruzione 138
Priorità 138
Interruzioni mascherabili e non mascherabili 141
3. **Le interruzioni nel microprocessore Z 80** 142
Interruzioni mascherabili 142
Interruzione non mascherabile 145
4. **Esempi software-hardware dei vari modi di interruzione nello Z 80** 145
Istruzione EI 146
Istruzione DI 146
Istruzione IM O 147
Istruzione RETI 147

Modo 1 153
Modo 2 154
Modo non mascherabile 155
5. Z 80 PIO (Parallel Input/Output Interface) 155
Architettura interna del PIO 155
Struttura interna di una porta 156
Descrizione dei registri 156
Descrizione dei segnali del PIO 157
6. Programmazione del PIO Z 80 159
Caricamento del vettore d'interruzione 160
Definizione del modo di funzionamento 161
Modo d'uscita o modo 0 162
Modo d'ingresso o modo 1 163
Modo bidirezionale o modo 2 164
Modo di controllo di bit o modo 3 164
Definizione del controllo delle interruzioni 165
Indirizzamento del PIO 166
Determinazione dell'indirizzo di I/O 167
7. Esempi di programmazione del PIO nei vari modi di funzionamento 168
8. DMA (Direct Memory Access) 171
9. Controllo di un sistema industriale di processo 172
10. Interfacciamento di un display 175
11. Interfaccia tra due sistemi a microprocessore 177
12. Interfaccia per il collaudo di circuiti integrati 180
13. Interfaccia di una tastiera 187
14. CTC Z 80 (Counter Timer Circuit) 194
15. Fan-Out e Fan-In nei sistemi a microprocessore 196
Esercizi proposti 197

CAPITOLO QUINTO **LA TRASMISSIONE ED IL RUMORE**

1. Generalità 199
2. Trasmissione asincrona 200
3. Trasmissione sincrona 202
Protocolli sincroni 203
Funzione dei protocolli sincroni 204
4. La rilevazione degli errori 205
Controllo di parità orizzontale 205
Controllo di parità verticale 205
Controllo ciclico ridondante o CRC (Cyclic Redundancy Checking) 206
5. Modo di trasmissione dell'informazione 207
- simplex 207
- semi-duplex 207
- duplex-completo 208
- eco-plex 208
6. Velocità di tramissione 208
7. I codici 209
Codice BCD (Binary Coded Decimal) 209
Codice Aiken 210
Codice Gray 210
Codice eccesso 3 210
8. USART (Universal Synchronous Asynchronous Receiver Transmitter) 213
9. Il modem 215
10. Interfacce digitali standard 215
EIA RS-232-C 216

Loop di corrente ... 220
11. Generatori di Baude-Rate ... 221
12. Rumore ... 223
Rimbalzi ... 224
13. Effetto delle riflessioni su un sistema digitale ... 230
14. Rumore sulle linee di alimentazione ... 233
15. Crosstalk ... 236
16. Alimentazione e disaccoppiamenti ... 240
17. I Bus per microcomputer ... 242
VERSAbus ... 243
MULTIBUS ... 244
MUBUS/EUROBUS ... 245
UNIBUS ed LSI-11 ... 248
S-100 bus ... 248
STD bus ... 250
18. Trasmissione delle informazioni mediante fibre ottiche ... 250
Componenti di una linea di trasmissione ottica ... 250
Parametri ottici ed elettrici ... 252
Tempi di salita e propagazione degli impulsi lungo una linea di trasmissione ottica ... 254
Cavi ottici ... 254
Collegamento di sistemi digitali per via ottica ... 255
Componenti digitali speciali per la trasmissione ottica delle informazioni ... 260
Esercizi proposti ... 265

CAPITOLO SESTO **APPLICAZIONI HARDWARE-SOFTWARE**

1. Acquisizione di dati da un bus ad otto linee ... 267
2. Contatore guidato a microprocessore ... 271
3. Acquisizione dati mediante microcomputer ... 279
4. Controllo di un braccio meccanico a più gradi di libertà ... 289
Progetto della scheda di interfaccia ... 296
Software ... 305
5. Conversione Analogico/Digitale ... 309
Software ... 311
6. Conversione Digitale/Analogica ... 312
Esercizi proposti ... 325

CAPITOLO SETTIMO **PROGETTO DI UN MICROCOMPUTER PER USI GENERALI**

1. Introduzione ... 327
2. La scelta del microprocessore ... 329
3. Il clock del sistema ... 334
Configurazione del circuito di reazione ... 337
Circuiti pratici ... 339
4. Il bus e la consolle ... 340
5. La memoria e la sua abilitazione ... 344
Rete di abilitazione della memoria ... 344
Rete di comando lettura/scrittura (R/W) ... 350
6. I comandi della consolle ... 351
- sezione dati ... 351
- sezione indirizzi ... 352
- sezione controlli ... 352
- sezione display ... 353
7. Dispositivo esterno ... 355
8. Impiego del microcomputer ... 356

Accensione 356
Memorizzazione di un programma 356
Esecuzione di un programma 356
9. Il sistema operativo o monitor 359

CAPITOLO OTTAVO **MEMORIE DI MASSA**

1. Memorie a scorrimento di carica o CCD (Charge Coupled Device) 365
Struttura interna di un SPS CCD 369
2. Memorie a bolle magnetiche 370
3. Nastri magnetici 374
4. Floppy Disk 375
Formattazione 378
Codifica dell'informazione 378
5. Sistemi vari di memorizzazione 379
Sistemi a laser 379
Sistemi a microfilm 380

CAPITOLO NONO **LA STRUMENTAZIONE DIGITALE**

1. Introduzione 381
2. Sonde logiche 382
Isolamento di un difetto in una zona 383
Isolamento di una porta logica difettosa 383
3. Analizzatori logici 384
Analizzatore di temporizzazioni 386
Analizzatore di stati logici 387
4. Analizzatori di firma 390
Firme corrette od errate 394
5. Sistemi di sviluppo 394
Funzionamento di un sistema di sviluppo 397

CAPITOLO DECIMO **APPLICAZIONI DEL SISTEMA MINIMO Z80**

1. Test di collaudo del circuito integrato SN 7446 401
Assembly 405
2. Test di collaudo del circuito integrato SN 7442 409
Assembly 412
3. Semplice applicazione di un sistema a microprocessore 414
Analisi Hardware 414
Analisi Software 415
Conclusioni 419
4. Semplice videogioco realizzato a microprocessore 421
Analisi Software 425
1. Inizializzazione 425
2. Programma principale 430
3. Routine di gestione dell'interruzione 430
4. Routine di delay 431

SOLUZIONI ESERCIZI PROPOSTI 433
GLOSSARIO 451
BIBLIOGRAFIA 467

CAPITOLO PRIMO

ELETTRONICA PROGRAMMABILE

1. Introduzione

Nel volume «Elettronica Digitale» sono state sviluppate le tecniche di base necessarie alla analisi e sintesi di reti combinatorie e sequenziali. Generalmente lo scopo di tale studio è quello di costruire dispositivi, chiamati anche controllori, atti a dirigere l'andamento delle operazioni di un qualsiasi sistema, sia esso di calcolo che industriale.

Il progetto dei controllori viene effettuato una volta che sia stato stabilito l'*algoritmo* e cioè *una successione finita di passi definiti, che permettono di risolvere una classe di problemi.* Questo significa che problemi simili, ma con dati diversi, vengono risolti utilizzando sempre il medesimo procedimento, vale a dire il medesimo algoritmo, anche se con un numero di operazioni diverse.

Successivamente si può procedere alla realizzazione pratica della circuiteria necessaria alla sua implementazione *hardware.* Con il nome di «*hardware*» si intende individuare tutti gli elementi materiali costituenti un elaboratore elettronico, vale a dire i circuiti ed i componenti elettronici e meccanici che lo compongono.

Naturalmente ogni volta che cambia il problema da risolvere, cambia anche l'algoritmo e quindi anche tutta la circuiteria. Questo approccio, tipicamente hardware, risulta valido ed economico solamente se tali cambiamenti sono in numero limitato nel tempo e se la complessità del circuito è minima.

Il maggior pregio di una esecuzione esclusivamente basata sull'hardware, è la elevatissima velocità di esecuzione che è limitata unicamente dai tempi di propagazione delle porte logiche impiegate, mentre il principale difetto è la necessità di dover sostituire il circuito ogni volta che cambia l'algoritmo.

Un passo avanti per il superamento di questa grave limitazione, può essere effettuato utilizzando memorie di tipo R0M (o PR0M) sotto forma di circuito combinatorio o sequenziale, oppure ricorrendo alle reti logiche programmabili PLA.

2. Uso di una memoria PROM come circuito combinatorio

Nel capitolo ottavo (paragrafo 8) del volume Elettronica Digitale, si è mostrato l'impiego di una memoria RAM come circuito combinatorio. In quel caso ad ogni interruzione sia pure momentanea dell'alimentazione, corrispondeva la cancellazione dei dati in essa immagazzinati con la conseguenza di non avere più un corretto funzionamento. Nei casi invece, nei quali si desidera una soluzione non volatile, si può ricorrere all'uso di una PROM che può essere scritta, mediante una operazione che viene chiamata *mascheratura,* direttamente dall'utilizzatore.

Se il numero di memorie da mascherare è elevato, tale operazione può essere effettuata direttamente dal costruttore durante il processo di fabbricazione, ottenendo così la sostituzione di una complessa interconnessione di porte logiche con il solo uso di una memoria. Per chiarire il modo di procedere, si farà uso di un esempio.

Si supponga di voler realizzare un circuito combinatorio ad otto variabili di ingresso, che generi le seguenti quattro funzioni di uscita:

$f_1 = A\,\overline{B}\,C\,D\,\overline{E}\,F\,G\,\overline{H}$

$f_2 = A\,\overline{B}\,C\,D\,\overline{E}\,F\,G\,H + \overline{A}\,\overline{B}\,C\,D\,E\,F\,\overline{G}\,H + A\,\overline{B}\,\overline{C}\,D + \overline{B}\,\overline{C}\,\overline{D}\,\overline{G} + \overline{A}\,B\,C\,\overline{D}\,E\,\overline{F}\,G\,H + B\,\overline{D}\,F\,\overline{G}\,H$

$f_3 = B\,\overline{D}\,F\,\overline{G}\,H$

$f_4 = A\,C\,\overline{D}\,E\,\overline{F}\,\overline{G}\,H + B\,\overline{C}\,D\,H + A\,B\,C\,D\,E\,\overline{F}\,\overline{G}\,\overline{H}$

Procedendo secondo le usuali regole, si otterrebbe il circuito combinatorio di figura 1 (non sono state effettuate le semplificazioni).

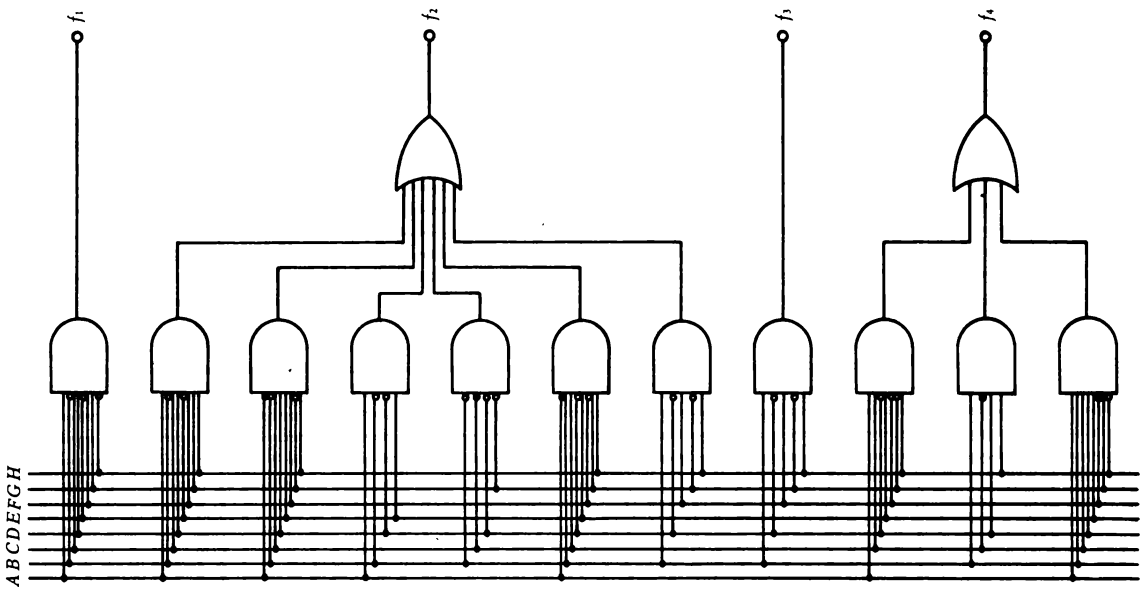


FIG. 1. - Circuito combinatorio ad otto ingressi e quattro uscite.

Nel precedente circuito, le uscite si porteranno al livello logico zero oppure uno secondo la particolare configurazione presente sugli ingressi. Ad esempio, se essi assumono la seguente configurazione:
A = 1, B = 1, C = 0, D = 0, E = 0, F = 1, G = 0, H = 1 le quattro funzioni avranno i valori:

$$f_1 = 0$$
$$f_2 = 1$$
$$f_3 = 1$$
$$f_4 = 0$$

Volendo utilizzare una memoria PR0M anziché il circuito combinatorio, basterà scrivere in ogni locazione di memoria individuata dalle otto variabili di ingresso, una parola di quattro bit corrispondente alla configurazione delle funzioni di uscita. Nel caso particolare dell'esempio trattato, occorre scrivere nella locazione di indirizzo 11000101 (n° 197) la parola 0110. La scelta della memoria deve cadere tra quelle aventi otto ingressi di indirizzo e con parallelismo di quattro, vale a dire una 256 × 4 come quella mostrata in figura 2.

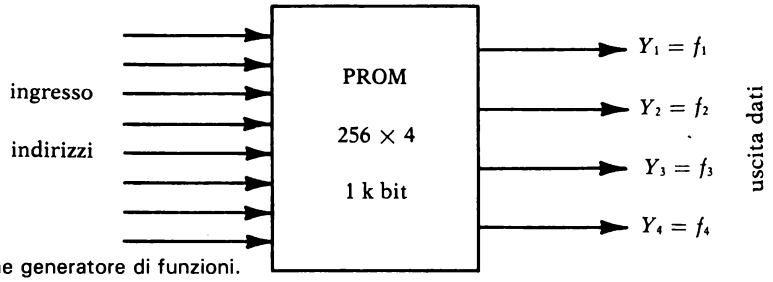


FIG. 2. -
PROM usata come generatore di funzioni.

La lista completa delle parole da scrivere nella PR0M (mascheratura), affinché per ogni combinazione delle variabili di ingresso si ottengano le rispettive funzioni di uscita, può ricavarsi dalla tabella della verità di figura 3, ottenuta come illustrato nel seguito per una funzione alla volta:

a) per ciascun termine della funzione, se contiene tutte le variabili, si scrive un 1 nella sua casella corrispondente. Ad esempio f_1 avrà un 1 solamente nella casella n° 173 cioè 10110110;
b) se in un termine mancano n variabili, occorre scrivere 2^n uno. Ad esempio per f_3 che ha solo cinque variabili occorre scrivere $2^3 = 8$ minterm, corrispondenti a tutte le locazioni di indirizzo X1X0X101.

Poichè una PR0M (o una R0M) può immagazzinare qualsiasi funzione, risulta chiara la sua grande versatilità per realizzare ogni tipo di circuito combinatorio come ad esempio la conversione di codici e la generazione di sequenze prestabilite.

In ogni caso, occorre tenere presente che, in determinate applicazioni per non elevare troppo il numero di ingressi o di uscite di una memoria, è bene fargli svolgere solo una parte del lavoro, demandando ad altri circuiti MSI l'esecuzione di semplici specifiche funzioni.

INGRESSI	GH = 00					GH = 01					GH = 10					GH = 11				
A B C D E F	Parole	Y_4	Y_3	Y_2	Y_1	Parole	Y_4	Y_3	Y_2	Y_1	Parole	Y_4	Y_3	Y_2	Y_1	Parole	Y_4	Y_3	Y_2	Y_1
0 0 0 0 0 0	0			1		64			1		128					192				
0 0 0 0 0 1	1			1		65			1		129					193				
0 0 0 0 1 0	2			1		66			1		130					194				
	3			1		67					131					195				
	4					68					132					196				
	5					69					133					197				
	6					70					134					198				
	7					71					135					199				
	8					72					136					200				
	9					73					137					201				
	10					74					138					202				
	11					75					139					203				
	12					76					140					204				
	13					77					141					205				
	14					78					142					206				
	15					79			1		143					207				
	16					80					144					208				
	17					81		1	1		145					209				
	18					82					146					210				
	19					83		1	1		147					211				
	20					84	1				148					212	1			
	21					85	1				149					213	1			
	22					86	1				150					214	1			
	23					87	1				151					215	1			
	24					88					152					216				
	25					89		1	1		153					217				
	26					90					154					218				1
	27					91		1	1		155					219				
	28					92					156					220				
	29					93					157					221				
	30					94					158					222				
	31				1	95					159					223				
	32				1	96				1	160					224				
	33				1	97				1	161					225				

						34		1		98			1	162			226						
						35		1		99			1	163			227						
						36		1		100			1	164		1	228			1			
						37		1		101			1	165		1	229			1			
						38		1		102			1	166		1	230			1			
						39		1		103			1	167		1	231			1			
						40				104				168			232						
						41				105				169			233						
						42				106	1			170			234						
						43				107				171			235						
						44				108				172			236						
						45				109				173		1	237			1			
						46				110				174			238						
						47				111				175			239						
						48				112				176			240						
						49				113		1	1	177			241						
						50				114				178			242						
						51				115		1	1	179			243						
						52				116	1			180			244	1					
						53				117	1			181			245	1					
						54				118	1			182			246	1					
						55				119	1			183			247	1					
						56				120				184			248						
						57				121		1	1	185			249						
						58				122	1			186			250						
						59				123		1	1	187			251						
						60				124				188			252						
						61				125				189			253						
						62			1	126				190			254						
1	1	1	1	1	1	63				127				191			255						

FIG. 3. - Tabella per la scrittura della PROM di fig. 1.

Reti logiche programmabili (PLA e FPLA)

I circuiti PLA (Programmable Logic Array) possono essere visti contemporaneamente come elementi di memoria, convertitori di codice o generatori di funzioni logiche.

Gli stati realizzati in uscita sono programmati dall'utilizzatore all'atto della costruzione consentendo un notevole risparmio di hardware, però l'elevato costo di produzione ne fa consigliare l'uso solo per elevati quantitativi.

Rispetto alle memorie ROM il principale vantaggio è costituito dal minor numero di livelli logici necessari per produrre la stessa funzione di controllo e ciò aumenta l'efficienza dinamica del sistema ove sono impiegati.

Dal punto di vista logico una rete PLA è costituita da un insieme di porte AND le cui uscite possono essere messe in OR tra di loro in modo da ottenere una somma di prodotti. In figura 4 è mostrato lo schema di questo dispositivo.

A differenza delle ROM, hanno un numero maggiore di ingressi poiché con un PLA essi non specificano solamente l'indirizzo dei bit in uscita, ma anche la funzione logica con la quale sono legati. Ad esempio la rete PLA di figura ha 14 ingressi ed 8 uscite e ciascuna di queste ultime è composta dall'OR di 96 porte AND con 14 ingressi. Mentre una ROM equivalente con 14 ingressi ed 8 uscite avrebbe una capacità di $2^{14} \times 8$ bit = 16.384 parole di 8 bit. In definitiva una PLA può essere trattata come una memoria di limitata capacità ed il profitto di una sua applicazione richiede che tutti i prodotti parziali necessari per il particolare impiego non superi il numero di 96.

ESEMPIO

Mediante l'uso di una rete PLA si vogliono ottenere le tre funzioni Y_1, Y_2 ed Y_3, che rispettino la seguente tabella della verità (tab. 1), in funzione dei quattro ingressi I_1, I_2, I_3 ed I_4.

TABELLA 1

I_1	I_2	I_3	I_4	Y_1	Y_2	Y_3
1	1	1	1	0	1	0
0	1	0	0	1	1	0
1	0	0	1	1	0	0
1	0	0	0	0	1	1
0	0	1	1	0	0	0
0	1	1	1	1	1	1
1	0	1	0	0	0	1

Come si può osservare, ogni funzione di uscita è composta dalla somma (0R) di un certo numero di prodotti canonici (AND) e cioè:

$$Y_1 = m_4 + m_9 + m_7$$
$$Y_2 = m_{15} + m_4 + m_8 + m_7$$
$$Y_3 = m_8 + m_7 + m_{10}$$

Per realizzare tali funzioni occorre disporre di una PLA con almeno quattro ingressi e tre uscite sulla quale realizzare le opportune connessioni.

Quando le necessità di impiego richiedono un numero limitato di reti da mascherare, si può ricorrere all'impiego delle FPLA (Field-Programmable Logic Array) che, a differenza delle PLA, possono facilmente essere programmate dal singolo utente.

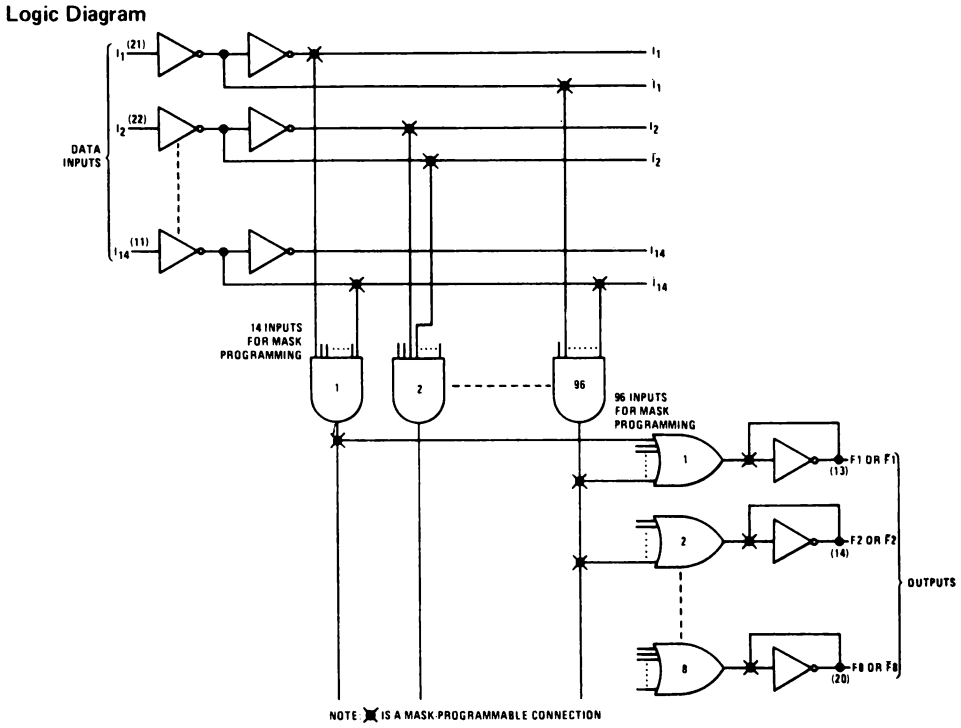


FIG. 4. - Schema semplificato di rete logica programmabile. (le X rappresentano le connessioni da effettuarsi mediante l'operazione di mascheratura).

3. Uso di una PROM come circuito sequenziale

Il modello strutturale di un circuito sequenziale, corrispondente ai modelli matematici di Mealy o di Moore, è già stato introdotto nel capitolo quinto del volume «Elettronica Digitale» ed è rappresentato nella figura 5.

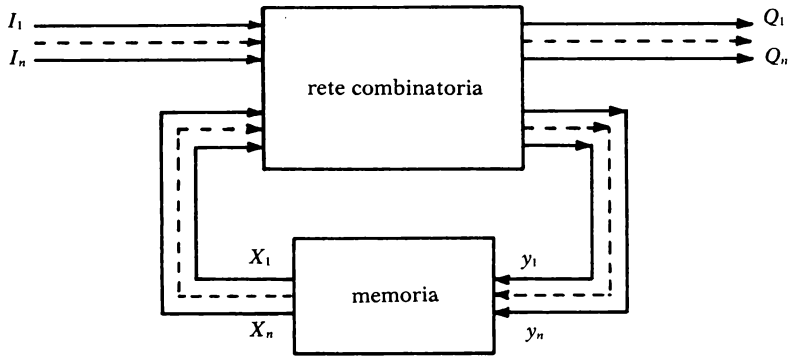


FIG. 5. - Modello generale di un circuito sequenziale.

Secondo tale struttura, in una rete sequenziale la successione degli stati di uscita è completamente definita allorché è assegnata la sequenza degli stati di ingresso e lo stato interno iniziale. Ciò è dovuto alla presenza di elementi di memoria (ad esempio registri e f-f) in un anello di reazione tra uscite ed ingressi.

Tutte le reti sequenziali possono pensarsi suddivise in:

a) *reti autonome:* sono quelle che, pur accordandosi con il modello di figura 5, non hanno ingressi I. Il solo segnale in arrivo è il clock del sistema, per cui la rete genera una sequenza di stati periodica (tranne una sequenza iniziale aperiodica) non potendosi intervenire in alcun modo dall'esterno.

Appartengono a questa categoria i contatori ed i registri a scorrimento. Queste reti possono essere impiegate come controllori di sistemi che ripetono sempre le stesse operazioni elementari.

b) *reti con ingressi esterni:* sono quelle che adeguano il loro funzionamento secondo le necessità indicate dagli ingressi I. Si ha così una sorta di «istruzione» inviata alla rete che ne consente l'interazione con l'esterno.

c) *reti con ingresso dal sistema:* in queste ultime, vengono riportate in ingresso delle informazioni ottenute dall'elaborazione stessa di quelle inviate al sistema.

La scelta sul tipo di memoria da utilizzarsi per la realizzazione di una qualsiasi di tali reti, dipende essenzialmente da tre fattori:

1) il parallelismo della memoria, vale a dire il numero di uscite dati, che deve essere almeno uguale al numero delle funzioni di uscita del circuito sequenziale da sostituire;

2) il numero delle linee di indirizzo che, come si vedrà dipende direttamente dall'algoritmo da eseguire;
3) il tempo di accesso, dal quale dipende la velocità di operazione e la dissipazione di potenza.

Se, ad esempio, si volesse realizzare un circuito divisore in frequenza per trentadue, usando una memoria anziché cinque f-f come mostrato nella figura 6, sarebbe necessario utilizzarne una con almeno cinque uscite

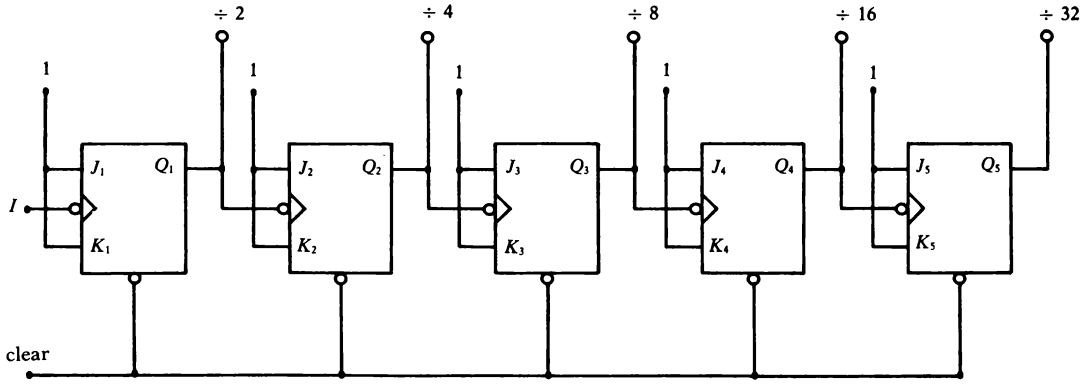


FIG. 6. - Divisore di frequenza per 32 (contatore a modulo 32) asincrono.

dati (pari al numero di f-f da sostituire), la quale potrà anche essere utilizzata per realizzare tutti i tipi di contatori a modulo minore di 32. Tutto ciò variando solamente i dati in essa immagazzinati, anziché cambiare di volta in volta il circuito.

La successione di «0» e di «1» che deve essere scritta nella PROM costituisce un primo semplice esempio di circuito programmabile e di programma.

Il metodo utilizzato per la realizzazione di questa tecnica verrà esposto parallelamente ad un esempio:

si supponga di voler progettare un divisore sincrono per otto, utilizzando una memoria PROM.

In accordo con lo schema di figura 5, alcune o tutte le uscite dati debbono essere rinviate in ingresso (indirizzi) per formare la configurazione base di un circuito sequenziale, come è mostrato in figura 7.

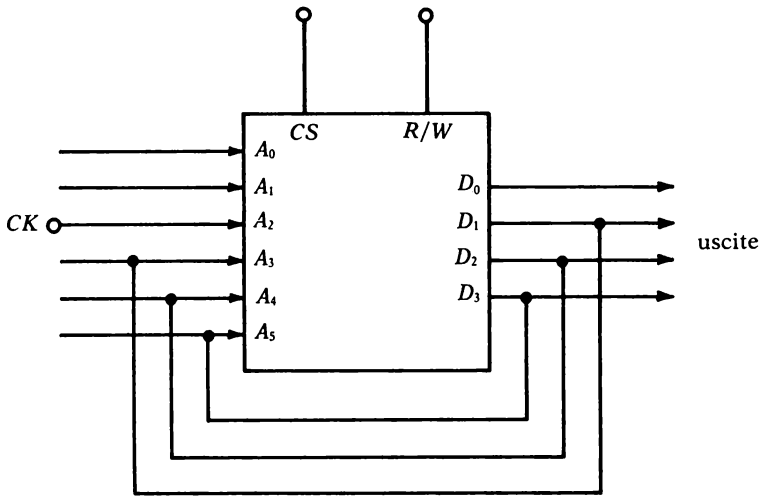


FIG. 7. - Circuito sincrono con retroazione.

A causa delle connessioni, per ogni stato delle uscite viene definito l'indirizzo di memoria ove sono depositati i dati che costituiranno le prossime uscite, dietro il comando di un clock esterno che sincronizza il funzionamento alla velocità desiderata.

Analogamente al modo di progetto dei contatori, si procede nella sequenza specificata dai seguenti passi;

a) disegno del diagramma temporale degli stati delle uscite;
b) realizzazione della tabella di flusso degli stati;
c) realizzazione della tabella delle eccitazioni;
d) realizzazione della tabella di programmazione o mascheratura.

a) disegno del diagramma temporale delle uscite

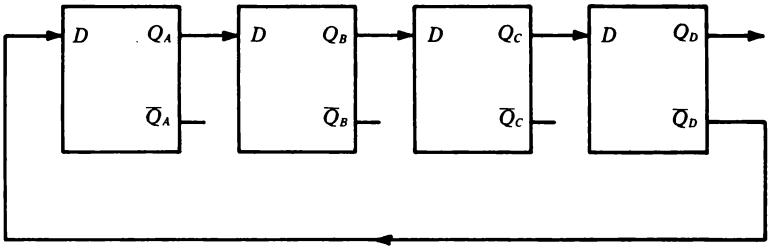
L'esecuzione del disegno ricalca gli schemi già noti per qualsiasi divisore. Nel caso presente è bene scegliere come istante di partenza quello in cui tutte le uscite sono al livello logico 1, che è lo stato a memoria disabilitata. In questo modo si obbliga il sistema a partire da una configurazione che sicuramente non sarà di blocco.

Inoltre, per evitare stati indesiderati che si potrebbero verificare a causa dei tempi di propagazione (alea statica), è bene imporre che cambi una sola uscita alla volta, di quelle interessate alla retroazione.

Un contatore che ha la caratteristica di cambiare un solo bit per ogni cambiamento di stato viene chiamato *contatore di Moebius.*

Nella figura 8 è mostrato un esempio di circuito di questo tipo, utilizzante flip-flop di tipo D.

Come si osserva dalla tabella della verità, l'uscita di ciascun f-f è uguale a quella del f-f precedente come avviene nei registri a scorrimento.



Q_A	Q_B	Q_C	Q_D	Stati binari
0	0	0	0	0
1	0	0	0	8
1	1	0	0	12
1	1	1	0	14
1	1	1	1	15
0	1	1	1	7
0	0	1	1	3
0	0	0	1	1

FIG. 8. - Contatore di Moebius a 4 bit e tabella degli stati.

Con un contatore di questo tipo, vengono generati otto stati e, più in generale, con n uscite si possono generare 2 n stati diversi.

Un ulteriore vantaggio dei contatori di Moebius è che ciascuno stato può essere codificato (riconosciuto) tramite una porta AND a due soli ingressi. Infatti basta considerare solamente l'1 e lo 0 adiacente (ad esempio 12, corrispondente a 1100, può decodificarsi con $Q_B \overline{Q}_C$, mentre 1 sarà uguale a $\overline{Q}_C Q_D$) poichè gli altri termini risultano «indifferenti», come può essere verificato mediante l'applicazione delle mappe di Karnaugh.

Da quanto detto, lo schema temporale è quello mostrato in figura 9.

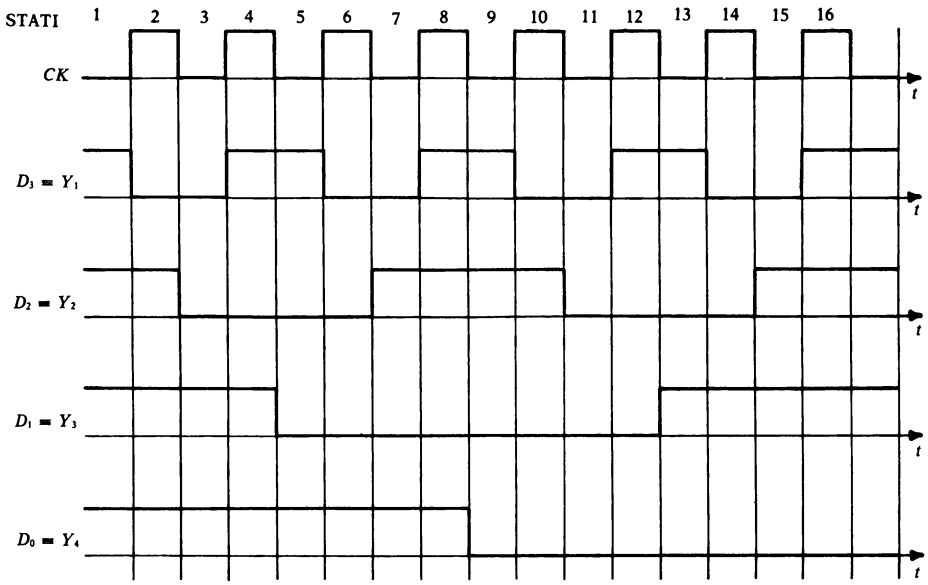


FIG. 9. - Uscite del divisore di frequenza per otto.

Con otto impulsi di clock, corrispondenti a sedici successivi salti di livello, si ottengono sedici stati diversi per cui sono necessarie quattro variabili ($2^4 = 16$) per rappresentarli tutti.

Queste ultime, sono state indicate nella figura con i simboli Y_1, Y_2, Y_3ed Y_4. (Si ricordi la ipotesi di far cambiare solo una uscita alla volta e si confronti questa figura con la fig. 7 del cap. 5 del volume Elettronica Digitale).

b) **tabella di flusso degli stati di uscita.**

Dal diagramma temporale del punto a), si può trarre la tabella di flusso di figura 10. Nelle due colonne di sinistra vengono riportati gli stati del clock che passa da zero ad uno o viceversa. I numeri stampati in neretto rappresentano gli stati stabili. Nelle colonne di destra sono indicate, con le variabili Y, le uscite della memoria che serviranno anche da retroazione verso l'ingresso.

Ad esempio la prima riga rappresenta lo stato stabile con il clock al livello zero e le uscite tutte ad uno, ma anche lo stato instabile che si ha quando il clock passa ad uno e le uscite non sono ancora cambiate.

CLOCK (stati)		USCITE				Stati binari
O	1	Y_1	Y_2	Y_3	Y_4	
1	2	1	1	1	1	15
3	**2**	0	1	1	1	7
3	4	0	0	1	1	3
5	**4**	1	0	1	1	11
5	6	1	0	0	1	9
7	**6**	0	0	0	1	1
7	8	0	1	0	1	5
9	**8**	1	1	0	1	13
9	10	1	1	0	0	12
11	**10**	0	1	0	0	4
11	12	0	0	0	0	0
13	**12**	1	0	0	0	8
13	14	1	0	1	0	10
15	**14**	0	0	1	0	2
15	16	0	1	1	0	6
1	**16**	1	1	1	0	14

FIG. 10. - Tabella di flusso degli stati di uscita.

c) tabella delle eccitazioni di uscita

La tabella delle eccitazioni di figura 11 è costituita da una mappa di Karnaugh a cinque variabili, poichè occorre considerare anche il clock tra le variabili di ingresso.

Le variabili di uscita sono solamente quattro (Y_1, Y_2, Y_3 ed Y_4) e quelle racchiuse da un circolo rappresentano gli stati stabili.

La sua compilazione viene effettuata con l'aiuto della tabella di flusso di figura 10 e risulta immediata. Per esempio, la prima riga della tabella di figura 10 individua due caselle nella mappa di Karnaugh e cioè $Y_1 = 1$, $Y_2 = 1$, $Y_3 = 1$, $Y_4 = 1$, $C = 0$ ed anche $Y_1 = 1$, $Y_2 = 1$, $Y_3 = 1$, $Y_4 = 1$, $C = 1$.

$Y_3 Y_4 C$ \ $Y_1 Y_2$	00	01	11	10
000	(0000)	0000	(1100)	1010
001	1000	(0100)	0100	(1000)
011	(0001)	1101	(1101)	0001
010	0101	(0101)	1100	(1001)
100	0110	(0110)	1111	(1010)
101	(0010)	1110	(1110)	0010
111	1011	(0111)	0111	(1011)
110	(0011)	0011	(1111)	1001

FIG. 11. - Tabella delle eccitazioni.

Nella prima casella, che individua lo stato stabile **1**, va collocato lo stato delle variabili **1111**, mentre nella seconda va collocato lo stato instabile 2 individuato dal successivo O111.

Continuando in questo modo, nella casella, $Y_1 = 0$, $Y_2 = 1$, $Y_3 = 1$, $Y_4 = 1$, $C = 1$, si metterà **0111** mentre nella casella $Y_1 = 0$, $Y_2 = 1$, $Y_3 = 1$, $Y_4 = 1$, $C = 0$ si metterà 0011 e così via fino al completamento della mappa.

d) tabella di programmazione

Dalla precedente elaborazione si può trarre la tabella di figura 12 che riporta per ogni indirizzo della PR0M i dati che debbono esservi immagazzinati (successione di zero ed uno).

Complessivamente occorrono 32 locazioni di memoria individuate da cinque linee di indirizzo che sono costituite dalle quattro uscite riportate in ingresso, più la linea del clock.

INDIRIZZI					DATI			
A_4	A_3	A_2	A_1	A_0	D_3	D_2	D_1	D_0
Y_3	Y_4	C	Y_1	Y_2	Y_1	Y_2	Y_3	Y_4
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	1	0
0	0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0	0
0	0	1	0	1	0	1	0	0
0	0	1	1	0	1	0	0	0
0	0	1	1	1	0	1	0	0
0	1	0	0	0	0	1	0	1
0	1	0	0	1	0	1	0	1
0	1	0	1	0	1	0	0	1
0	1	0	1	1	1	1	0	0
0	1	1	0	0	0	0	0	1
0	1	1	0	1	1	1	0	1
0	1	1	1	0	0	0	0	1
0	1	1	1	1	1	1	0	1
1	0	0	0	0	0	1	1	0
1	0	0	0	1	0	1	1	0
1	0	0	1	0	1	0	1	0
1	0	0	1	1	1	1	1	1
1	0	1	0	0	0	0	1	0
1	0	1	0	1	1	1	1	0
1	0	1	1	0	0	0	1	0
1	0	1	1	1	1	1	1	0
1	1	0	0	0	0	0	1	1
1	1	0	0	1	0	0	1	1
1	1	0	1	0	1	0	0	1
1	1	0	1	1	1	1	1	1
1	1	1	0	0	1	0	1	1
1	1	1	0	1	0	1	1	1
1	1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	1	1

FIG. 12. Tabella di programmazione.

In definitiva, volendo realizzare praticamente il divisore sincrono, occorre scegliere una memoria PR0M (oppure ROM) con almeno cinque ingressi di indirizzo e quattro ingressi dati ed effettuare la mascheratura scrivendo all'indirizzo 00000 il dato 0000, all'indirizzo 00001 il dato 0000 e così via fino ad arrivare all'indirizzo 11111 dove verrà scritto 0111.

Successivamente collegando D_0 con A_3, con D_1 con A_4, D_2 con A_0, D_3 con A_1 ed il clock ad A_2, ad ogni transizione di quest'ultimo si otterranno le configurazioni di uscita mostrate in figura 9.

4. Struttura di un microcomputer

Nel precedente paragrafo è stato introdotto, in modo non del tutto esplicito, il concetto di programmazione costruendo un sistema il cui schema a blocchi è rappresentabile nel modo mostrato dalla figura 13.

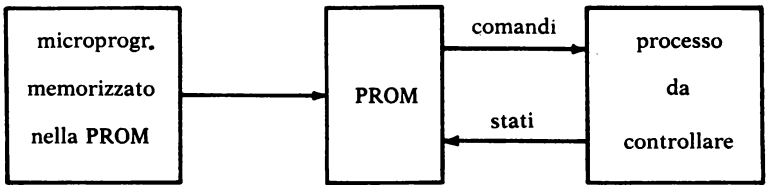


FIG. 13. - Semplice sistema a microprogramma.

Il blocco indicato con «microprogramma» è costituito dalla sequenza delle parole, formate da zero ed uno, che sono state scritte nella memoria per far eseguire un determinato algoritmo (la divisione per otto in frequenza). Volendo cambiare algoritmo (per esempio dividere per quattro) è necessario cambiare solo la successione ed il contenuto di tali parole.

Più in generale un programma viene definito come una sequenza di istruzioni, messe in un determinato ordine, che definiscono un algoritmo.

Compito delle istruzioni è quello di agire sui circuiti del sistema per far eseguire tutte le operazioni specificate nella opportuna sequenza. Tali direttive prendono il nome di *software*. Con questo nome si intende indicare genericamente l'insieme non materiale di un elaboratore elettronico.

I risultati della interazione fra *hardware* e *software*, costituiti da dati ed informazioni, verranno poi messi in comunicazione con l'esterno mediante opportuni circuiti di interfaccia detti di Input/Output (I/O).

Esempi comuni di dispositivi di ingresso sono:
- tastiere
- lettori di schede perforate, nastri magnetici, dischi magnetici e caratteri ottici.
- microfoni
- trasduttori di pressione, temperatura, ecc.
- fotocellule
- interruttori.

Mentre tra i dispositivi di uscita si possono elencare:
- display digitali e video
- schede perforate, nastri e dischi magnetici
- stampanti
- altoparlanti
- motori
- relays
- lampade e diodi LED.

Uno schema a blocchi che leghi software ed hardware per realizzare un unico sistema funzionale risulta con la costruzione di figura 14, la quale coincide perfettamente con quella di un microcomputer, cioè con quella di un elaboratore per usi generali (general purpose) basato su un'unità centrale LSI detta *microprocessore*.

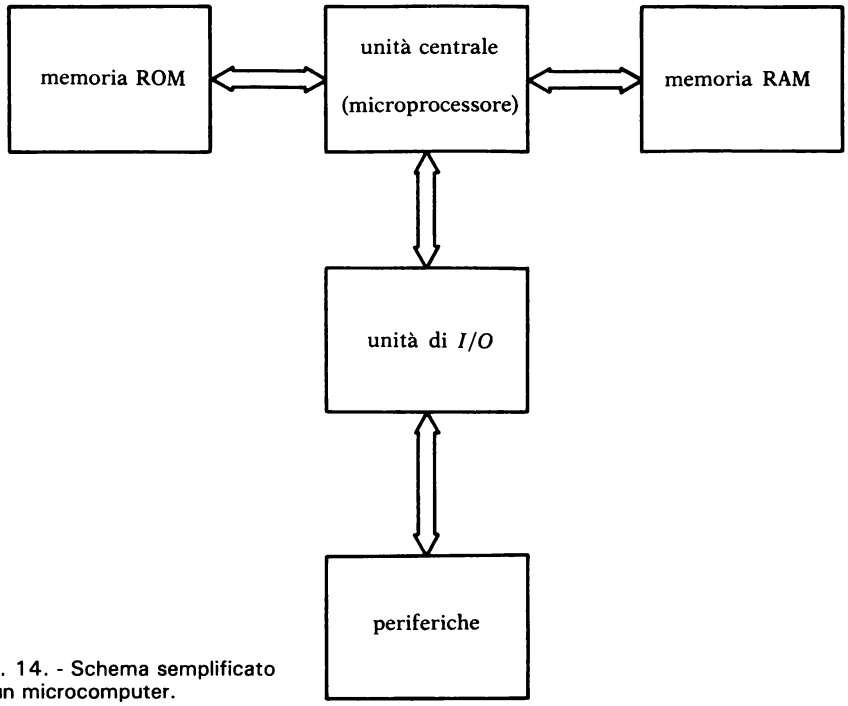


FIG. 14. - Schema semplificato di un microcomputer.

Per maggior chiarezza risulta necessario definire il significato di alcuni dei nomi introdotti:
COMPUTER: dispositivo di uso generale che esegue operazioni predefinite su informazioni in ingresso, fornendo determinati risultati. Con questo nome, solitamente, si indicano quelli di costo e dimensioni elevate che usano linguaggi di programmazione simili a quello dell'uomo e lavorano con parole composte da 32 o 64 bit.
MINICOMPUTER: indica un computer, anch'esso di uso generale, ma di dimensioni fisiche e costo più limitati. Usa normalmente parole di 16 o 32 bit.
MICROCOMPUTER: è il più piccolo della famiglia dei computer; usa parole di otto bit ed ha possibilità di memoria limitate.
MICROPROCESSORE: con questo nome viene indicata una unità centrale che è stata realizzata su un unico chip integrato.

Nella precedente figura, si è introdotta una suddivisione fra memoria ROM e memoria RAM che risulta necessaria poiché nei microcomputer «general purpose» varia di volta in volta il programma da eseguire secondo le necessità dell'utente (programma-utente).

Quest'ultimo viene perciò caricato nella memoria RAM, mentre nella ROM (non volatile) viene memorizzato all'atto della costruzione, il programma necessario alla gestione interna dei vari chip costituenti il microcomputer e che prende il nome di *Monitor*. Questo programma viene eseguito per primo all'atto dell'accensione del sistema per inizializzarlo, cioé per metterlo in condizione di attendere dei comandi, e successivamente esegue i sottoprogrammi delle istruzioni impartite.

Il verso delle frecce dei canali di connessione tra i blocchi, sta ad indicare genericamente in quale direzione viaggia il flusso delle informazioni.

Lo schema di figura 14 ricalca molto fedelmente la struttura ideata da Von Neumann che nel 1945 a Princeton, progettò quello che viene ritenuto il prototipo degli attuali elaboratori elettronici che si fondano sul concetto di *programma memorizzato.*

Nella struttura di Von Neumann, in un'unica memoria venivano immagazzinate in forma binaria, sia le istruzioni che i dati sui quali operare, ed in base alle necessità l'elaboratore era in grado di saltare da un'istruzione ad un'altra secondo decisioni elementari che poteva prendere.

Quando, invece, il sistema a microprocessore viene utilizzato in applicazioni industriali nelle quali viene eseguito in modo ripetitivo lo stesso compito, la precedente suddivisione fisica tende a scomparire poiché si preferisce caricare anche il programma utente in una memoria non volatile. In quest'ultimo caso il programma risulta protetto da qualsiasi disturbo accidentale quale ad esempio la momentanea interruzione dell'alimentazione.

L'unità di ingresso-uscita, svolge il compito di interfacciare il microprocessore a tutti quei sistemi atti al suo colloquio con il mondo esterno.

Attualmente i più diffusi micropr. sul mercato quali ad esempio lo Z-80

della Zilog, l'8080 della Intel, l'MC 6800 della Motorola, il TMS 990 della Texas I., ecc. adottano un parallelismo ad 8 bit. Ciò significa che sono strutturati, sia internamente che verso l'esterno, per lavorare con parole di otto bit di lunghezza (bytes). (Vedi tabella 2).

Il contenuto delle parole rappresenta sia i dati che gli indirizzi di memoria o le istruzioni o qualsiasi carattere del codice usato. Per esempio il byte 10000001 può rappresentare sia il numero 10000001_2 in binario che il numero 201_8 in ottale, oppure l'istruzione ADD A, C o l'indirizzo di una specifica locazione di memoria, secondo il particolare uso che se ne fa.

microprocess.	dimensione parola	n° registri interni	f (MHz) max	fasi di clock
8080A	8	7	2,6	2
8048	8	9	2	1
Z80A	8	14	4	1
MC 6800	8	4	2	2
2650	8	7	2	1
6502	8	VAR	4	1
SC/MP	8		4	1

Tabella 2. - Caratteristiche di alcuni microprocessori ad otto bit.

I primi nati usavano parole di quattro bit e set di istruzioni limitato ad un numero inferiore a dieci, mentre attualmente si stanno utilizzando microprocessori con parallelismo 16 od anche 32, e set di istruzioni in numero di qualche centinaia. Ciononostante hanno tutti una struttura a blocchi uguale a quella di figura 14, sebbene vari notevolmente l'organizzazione interna di tali blocchi da costruttore a costruttore.

I principali vantaggi dei sistemi basati sui microprocessori rispetto a quelli a logica cablata (hard-wired logic), sono:
a) basso costo, dovuto all'elevato numero di chip «general purpose» che possono essere prodotti e validi per le più disparate applicazioni;
b) basso numero di componenti ed elevata affidabilità;
c) elevatata versatilità.

Per contro, c'è una diminuzione della velocità di risposta rispetto alla logica cablata, dovuta alla necessità di dover eseguire delle istruzioni prima di generare i segnali di uscita. In linea generale sono quattro le caratteristiche che influenzano la velocità di lavoro di una CPU.

a) il *tempo di ciclo:* è definito come il tempo necessario per eseguire una istruzione base. Esso è costituito da una sequenza di stati più piccoli interni al microprocessore.

Aumentando il numero di tali passi si può diminuire la complessità dell'hardware, ma contemporaneamente aumenta la durata del tempo di ciclo.

b) la *tecnologia utilizzata;* le CPU a CM0S sono in genere più lente di quelle a TTL.

c) *l'ampiezza della parola;* al suo aumentare aumenta anche la velocità di lavoro poiché, ad esempio, una istruzione a 16 bit esegue un lavoro per il quale occorrerebbero più istruzioni ad otto bit.

d) il *set di istruzioni;* è definito come l'insieme di tutte le istruzioni che possono essere eseguite dal microprocessore. Il loro numero e la potenza è direttamente legato all'ampiezza della parola e all'efficienza della codifica, per cui due micro con parole di uguale ampiezza possono avere set di istruzioni molto diversi.

Nella figura 15 è mostrata l'area di applicazione di sistemi a microprocessore, in un grafico che riporta i costi di produzione rispetto al volume dei sistemi prodotti.

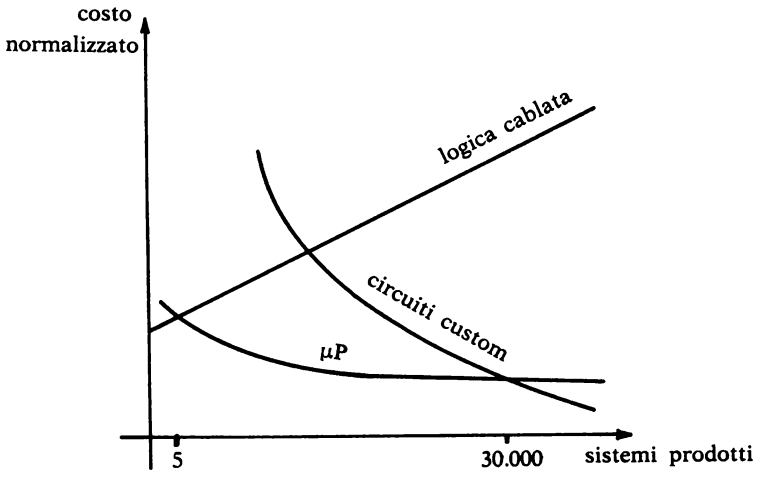


FIG. 15. - Campo di applicazione dei sistemi a microprocessore, per sistemi di complessità equivalente a 500 porte logiche (Motorola).

5. L'hardware del microprocessore

Il microprocessore, chiamato anche CPU (Central Processing Unit) o MPU (Micro Processing Unit) realizza il controllo e la manipolazione dei dati dell'intero microcomputer, e in esso risiede circa il 70% della potenza di elaborazione.

Lo schema a blocchi dei principali elementi costituenti la sua architettura interna, pur variando da costruttore a costruttore o secondo le qualità che si vogliono illustrare, assume la forma illustrata in figura 16.

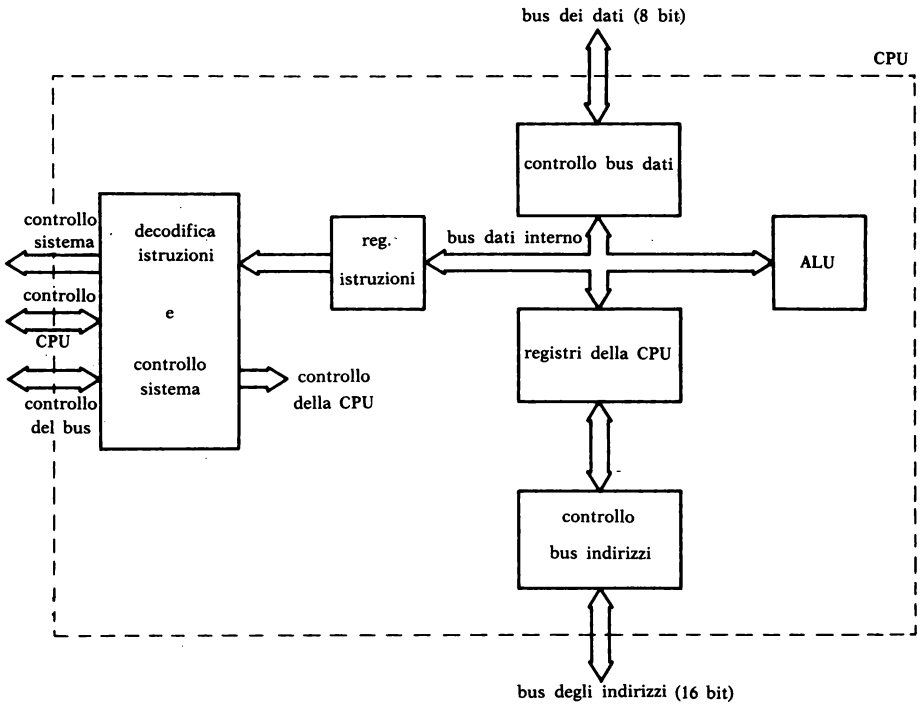


FIG. 16. - Schema a blocchi di una CPU (Zilog, SGS).

Come si vede, la CPU colloquia con l'esterno e tra gli stessi blocchi interni, mediante un sistema di collegamenti (fili) chiamato bus. Esso può essere concettualmente suddiviso in tre tipi:

1) bus degli indirizzi, sul quale viaggiano segnali che costituiscono tutto o parte di un'indirizzo; può essere a 4, 8, 12, 16 o 20 bit di parallelismo.

2) bus dei dati, sul quale viaggiano segnali che costituiscono i dati sui quali operare; è ad 8 o a 16 bit.

3) bus dei controlli, sul quale viaggiano segnali che controllano la temporizzazione del sistema (4 o più fili).

Il nome di bus sta ad indicare che, mettendo in comunicazione più blocchi diversi, può essere utilizzato di volta in volta solo da quelli interessati allo scambio di informazioni. Se esse viaggiano in un solo verso, il bus viene chiamato monodirezionale, e la freccia indica una sola direzione, mentre viene indicato come bidirezionale nel caso opposto.

Il numero dei fili che lo costituiscono è pari al numero di bit che vengono scambiati in parallelo tra i vari componenti del sistema.

Nella figura 17 è mostrato un esempio di struttura a bus.

Occorre sottolineare che per collegamenti di questo tipo, i singoli moduli devono essere di tipo tri-state, per consentire la loro *virtuale sconnessione* allorché non sono interessati all'informazione che in quell'istante viaggia sul bus comune.

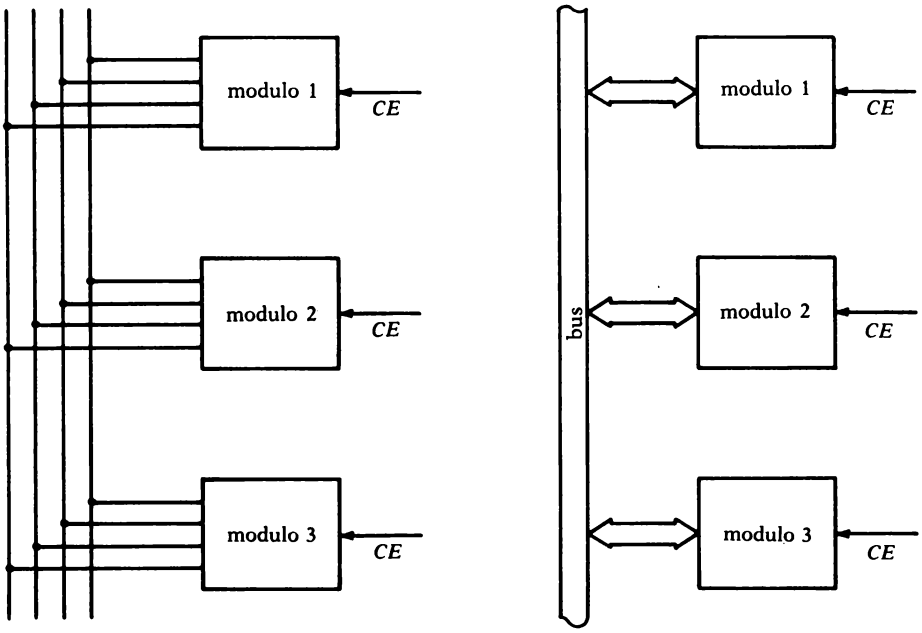


FIG. 17. - a) struttura a bus; b) schematizzazione di una struttura a bus.

Registro delle istruzioni e logica di controllo di microprogramma

Nel registro delle istruzioni viene caricata, di volta in volta, l'istruzione letta dalla memoria e identificata dal suo codice operativo (vedi capitolo 2). Il blocco di controllo del sistema provvede successivamente alla sua decodifica ed alla generazione della sequenza di operazioni elementari (microprogramma) per l'esecuzione di tale istruzione, emettendo tutti i segnali di controllo necessari e diretti sia verso l'interno che verso l'esterno della CPU.

Ciascuna operazione elementare del microprogramma è detta *microistruzione* ed è immagazzinata in una memoria ROM *interna al microprocessore,* all'atto della sua costruzione.

Quindi, nella memoria ROM interna sono immagazzinati tutti i microprogrammi relativi a tutte le istruzioni eseguibili dal particolare microprocessore.

Unità aritmetico-logica (ALU)

Questo blocco effettua tutte le elaborazioni aritmetiche e logiche all'interno della CPU, che possono suddividersi nel modo seguente:
- operazioni aritmetiche di somma e sottrazione
- operazioni logiche AND, OR, XOR
- scorrimenti e rotazioni di bit verso destra o sinistra
- comparazioni.

I risultati di tali operazioni sono sempre riportati in un particolare registro detto accumulatore.

Registri della CPU

I registri possono essere immaginati come particolari locazioni di memoria RAM situate all'interno della CPU, per essere usate nella manipolazione temporanea di dati ed indirizzi. Così facendo, non è necessario ricorrere continuamente all'uso della memoria esterna e quindi si ha una maggiore semplificazione delle operazioni oltre che un risparmio di tempo.

Il numero e la funzione dei registri può variare da micro a micro, ma generalmente le definizioni si adattano bene per ciascuno di essi.

Nella figura 18 è mostrata la configurazione dei registri della CPU Z 80 della Zilog (o SGS per l'Europa).

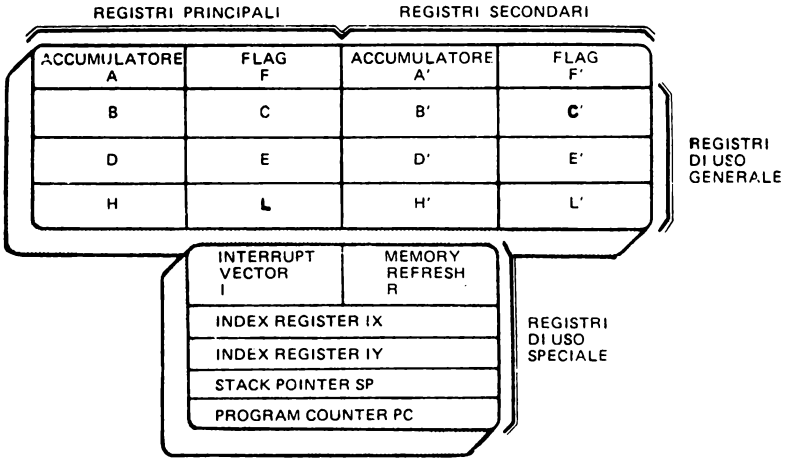


FIG. 18. - Configurazione dei registri della CPU Z-80.

Essi possono essere classificati come registri di uso speciale oppure di uso generale.

I registri di uso speciale sono:

PROGRAM COUNTER (PC); è a 16 bit, esso memorizza l'indirizzo della istruzione che è in corso di lettura dalla memoria e che deve essere eseguita. Viene automaticamente incrementato non appena il suo contenuto è posto sul bus degli indirizzi.

Più precisamente, durante la fase di prelevamento dell'istruzione (fase di fetch), il PC contiene il suo indirizzo; al termine di questa fase, quando l'istruzione viene eseguita (fase di execute), il PC viene incrementato e contiene così l'indirizzo della successiva locazione di memoria ove è collocata la prossima istruzione.

Se il programma prevede dei salti, il nuovo indirizzo da raggiungere viene automaticamente inserito nel PC durante la fase di execute senza eseguire l'incremento di uno.

STACK POINTER (SP); per stack si intende una qualsiasi parte della memoria RAM esterna nella quale vengono memorizzati, su richiesta del programmatore, i dati provenienti da un qualsiasi registro della CPU o viceversa. Lo stack pointer è un registro a 16 bit che memorizza l'indirizzo dell'ultimo dato inserito.

È organizzato secondo un sistema chiamato LIFO (Last-In-First-Out) secondo il quale l'ultimo dato scritto è anche il primo ad essere letto, come accadrebbe ad una pila di oggetti messi uno sull'altro.

INDEX REGISTER (IX ed IY); questi due registri di 16 bit ciascuno, vengono utilizzati per memorizzare un indirizzo base da usarsi quando si impiega un particolare metodo di indirizzamento detto *indicizzato*, come si vedrà nel prossimo capitolo.

INTERRUPT REGISTER (I); il registro di indirizzo per gli interrupt è ad 8 bit e contiene la parte più significativa dell'indirizzo del programma di gestione di una interruzione. La parte meno significativa (gli altri 8 bit) dell'indirizzo, è fornita dal dispositivo esterno che richiede l'interruzione della normale esecuzione del programma.

MEMORY REFRESH REGISTER (R); questo registro non viene quasi mai utilizzato dal programmatore. Agisce come contatore di rinfresco delle memorie dinamiche, incrementando automaticamente sette dei suoi otto bit dopo ogni lettura di una istruzione.

L'ottavo bit rimane fisso al valore dato tramite l'istruzione LD R,A.

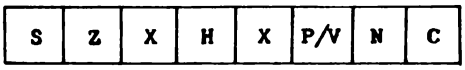
I registri di uso generale sono divisi in due gruppi perfettamente intercambiabili, che possono essere selezionati dal programmatore. Fra di questi, due rivestono particolare importanza e sono l'*accumulatore A* ed il *registro di FLAG*.

Come già esposto in precedenza, l'accumulatore memorizza automaticamente il risultato di qualsiasi operazione aritmetica o logica ad otto bit,

oltre a venire eventualmente caricato con dati su comando del programma. Inoltre è quasi sempre interessato dal lavoro della CPU.

Il registro di FLAG invece, contiene otto bit formanti una *parola di stato*. I singoli bit della parola, sono posti a zero oppure ad uno secondo il risultato di operazioni svolte generalmente dalla ALU.

Il loro nome è mostrato nella figura 19 (X indica un bit non significativo).



C = flag di carry
N = flag di sottrazione
P/V = flag di parity/overflow
H = flag di half/carry
Z = flag di zero
S = flag di segno

FIG. 19. - Struttura del registro di flag della CPU Z-80.

6. Segnali della CPU

La CPU ha generalmente una struttura Dual In Line (DIL), a 40 piedini per i sistemi ad otto bit, ed a 64 piedini per i sistemi a 16 bit, che sono disponibili per l'hardware esterno (memorie, circuiti di I/O ecc.).

La loro funzione può essere individuata in uno dei quattro raggruppamenti seguenti:

- bus dei dati
- bus degli indirizzi
- bus dei controlli
- alimentazione e clock

Non sempre però è rispettata una suddivisione così rigida poiché per limitare il numero dei piedini a volte su uno stesso bus viaggiano prima gli indirizzi e poi i dati, oppure sul bus dei dati viaggia solo la parte dei bit meno significativa degli indirizzi, ecc.

Il bus dei controlli, anche se porta questo nome, non ne svolge la funzione poiché ogni singolo filo è specializzato per un determinato segnale.

Nella figura 20 è mostrata la configurazione dei pin di alcuni microprocessori ad 8 bit che mette in evidenza una discreta varietà sia nel numero che nella funzione svolta da ciascun controllo.

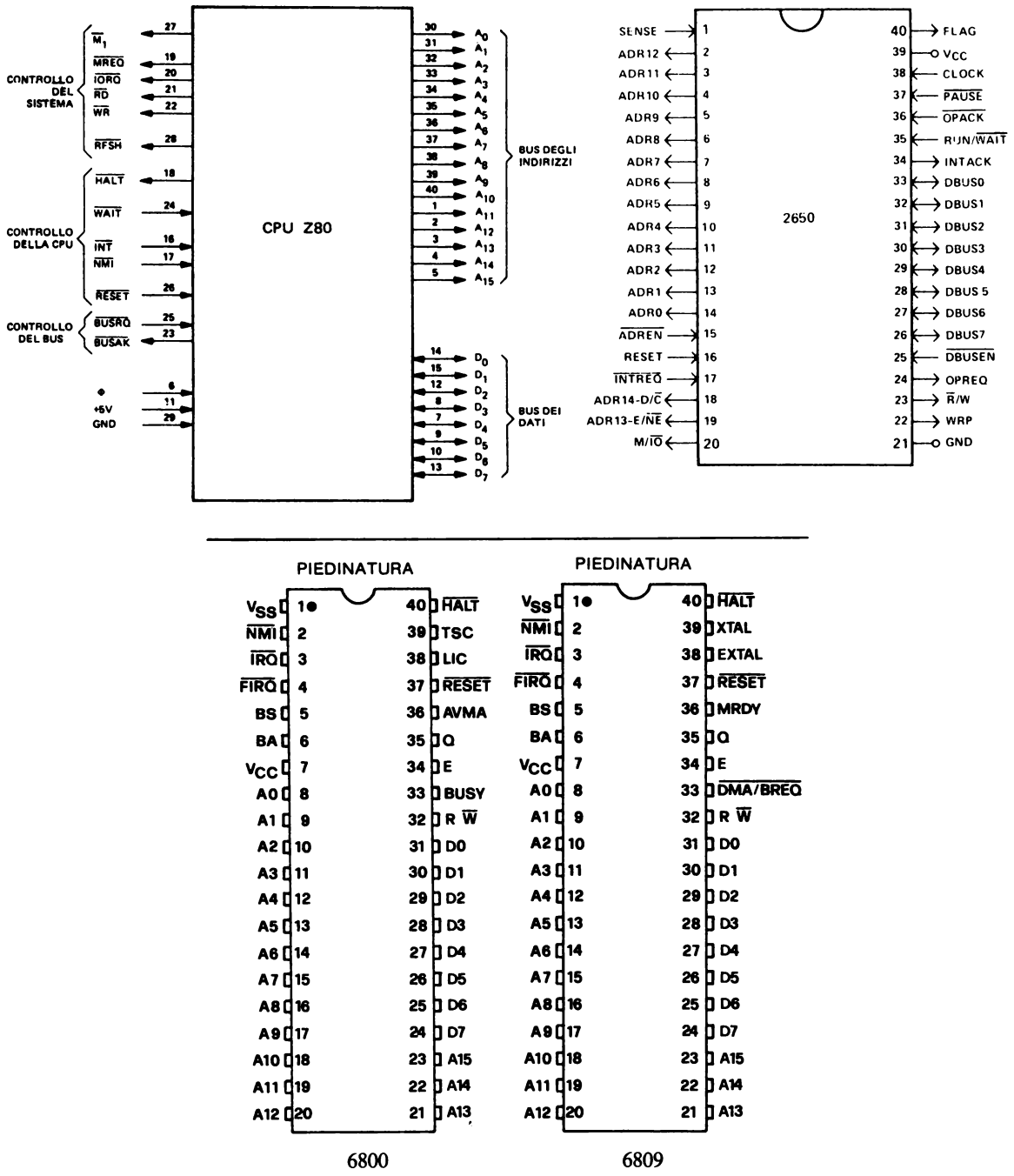


FIG. 20. - Pin-Out di alcuni microprocessori ad otto bit.

Occorre infine osservare che, per particolari e ben definite applicazioni si può ricorrere all'applicazione di micromputer «*single chip*» oppure ai «*bit-slice*».

I primi hanno la particolarità di ospitare in un unico chip tutte le parti costituenti un vero e proprio microcomputer (CPU, RAM, ROM, ecc.) come ad esempio l'8048 della Intel mostrato in figura 21.

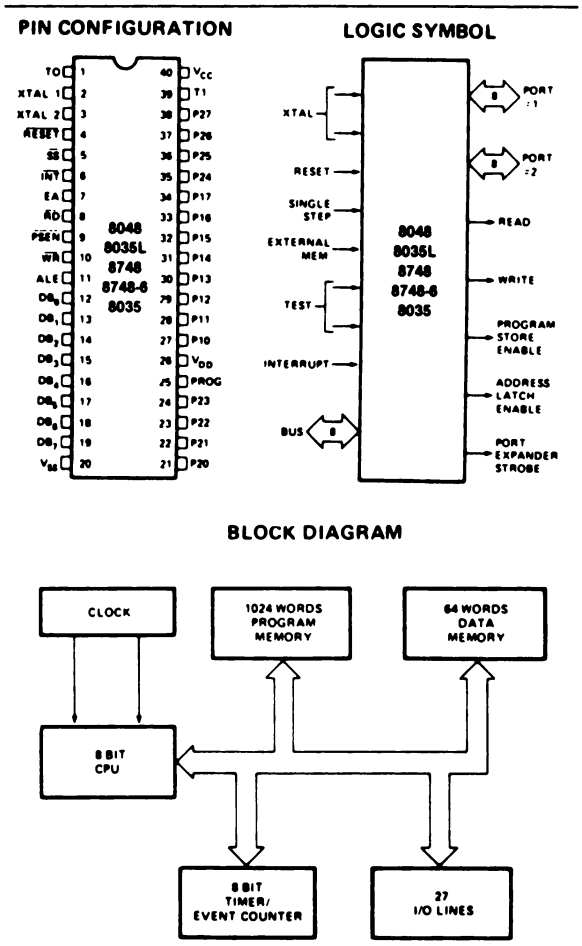


FIG. 21. - Microcomputer su singolo chip 8048 (Intel); piedinatura, descrizione funzionale e schema a blocchi.

Essi sono utilizzati soprattutto nelle applicazioni di alta tiratura e che necessitano di un basso costo, mentre non sono molto adatti per impieghi con elevate prestazioni.

Quando invece, è necessario unire una elevata velocità di operazione ad una grande mole di elaborazione dei dati, si può ricorrere all'uso dei microprocessori *bit-slice*.

Esempi di impiego si hanno nei radar, nella robotica, nella elaborazione sofisticata di immagini e nel controllo dei processi industriali.

I vantaggi derivanti dalla loro utilizzazione possono riassumersi in due direttrici principali:

1°) espandibilità, per cui si può scegliere la lunghezza della parola ponendo in cascata un certo numero di elementi bit-slice. Per esempio quattro elementi di una serie a due bit, costituiscono una CPU ad otto bit. Oppure si può variare il set di istruzioni secondo le esigenze dell'utente.

2°) velocità, poiché sono in tecnologia bipolare essa aumenta da 5 a 10 volte per ogni microciclo, rispetto alla tecnologia MOS.

Nella figura 22 è illustrata l'interconnessione di tre 2901 per formare una CPU da 12 bit.

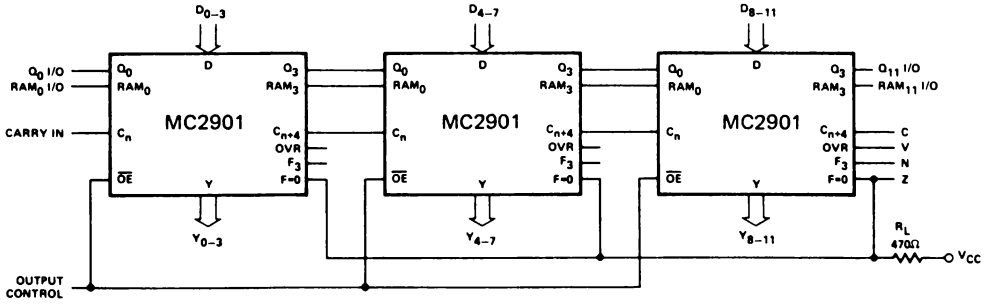


FIG. 22. - Tre bit-slice 2901 utilizzate per costruire una CPU da 12 bit.

Poiché nel seguito la maggior parte degli esempi, sia hardware che software, verranno eseguiti con l'ausilio dello Z-80 viene data qui di seguito la descrizione funzionale dei suoi singoli piedini.

Per non appesantire la trattazione, si è volutamente ridotta al minimo la parte descrittiva per cui il significato di alcuni segnali provenienti o diretti alla CPU potrà essere non del tutto chiaro, ma tale eventuale lacuna verrà colmata man mano che si procederà nel loro impiego.

D_0-D_7: bus dei dati costituito dagli otto bit di un dato; è bidirezionale, attivo al livello alto e tri-state.

A_0-A_{15}: bus degli indirizzi costituito dalle 16 linee necessarie all'indirizzamento di tutti gli eventuali 64 K byte di memoria. Gli otto bit meno significativi (A_0-A_7) sono utilizzati per la selezione di un massimo di $2^8 = 256$ diversi dispositivi di ingresso/uscita, mentre durante un'operazione di

rinfresco della memoria dinamica i sette bit meno significativi contengono un indirizzo di rinfresco valido.

È di tipo tri-state ed attivo al livello alto.

CONTROLLO DEL BUS

$\overline{\text{BUSRQ}}$: (bus request), quando questo segnale diviene attivo, la CPU pone il bus dei dati, il bus degli indirizzi e tutti i segnali di controllo con uscite tri-state, in alta impedenza. In questo modo il controllo su tali bus passa ad altri dispositivi che lo richiedono.

È attivo al livello basso.

$\overline{\text{BUSAK}}$: (bus acknowledge) è il segnale di risposta della CPU ai dispositivi richiedenti il controllo di bus, per indicare che sono stati posti in alta impedenza.

CONTROLLO DEL SISTEMA

$\overline{\text{M1}}$: (primo ciclo macchina) è un segnale emesso dalla CPU per indicare che il ciclo macchina in esecuzione è il ciclo di lettura di un codice operativo. Se il codice operativo di una istruzione è a due byte, il segnale M1 viene generato due volte.

Durante il ciclo di riconoscimento di una interruzione, M1 viene emesso insieme al segnale $\overline{\text{IORQ}}$.

$\overline{\text{IORQ}}$: (Input/output request) è un segnale emesso dalla CPU per indicare ai dispositivi esterni interessati che sugli otto bit meno significativi del bus indirizzi, è presente un indirizzo valido per una operazione di lettura o scrittura. L'uscita, di tipo tri-state, è attiva al livello basso.

$\overline{\text{RD}}$: (read), questo segnale indica la richiesta della CPU di leggere un dato da un dispositivo esterno o dalla memoria. L'uscita, di tipo tri-state, è attiva al livello basso.

$\overline{\text{WR}}$: (write), questo segnale indica alla memoria o ad altri dispositivi esterni che la CPU ha inviato sul bus un dato valido. L'uscita tri-state è attiva bassa.

$\overline{\text{MREQ}}$: (memory request), indica alla memoria che sul bus degli indirizzi è presente un indirizzo valido per una operazione di lettura o scrittura. L'uscita tri-state è attiva bassa.

CONTROLLO DELLA CPU

$\overline{\text{INT}}$: (interrupt request), la richiesta di interruzione proveniente da un dispositivo di I/O, viene soddisfatta al termine dell'istruzione che la CPU sta eseguendo, se BUSRQ non è attivo.

L'accettazione della interruzione è riconosciuta dall'emissione contemporanea dei segnali $\overline{\text{M1}}$ e $\overline{\text{IORQ}}$.

Il segnale $\overline{\text{INT}}$ è attivo basso.

$\overline{\text{NMI}}$: (non maskable interrupt), è un segnale di richiesta di interruzione con priorità rispetto ad $\overline{\text{INT}}$. Il segnale $\overline{\text{NMI}}$ costringe la CPU a ripartire (restart) dall'indirizzo 0066H, ma salva il contenuto del PC nello stack per poter successivamente ripartire dal programma interrotto.

$\overline{\text{HALT}}$: (stato di alt); è un segnale in uscita dalla CPU, che indica l'attesa di una interruzione mascherabile o non mascherabile. È attiva al livello basso.

$\overline{\text{WAIT}}$: (attesa); questo segnale di ingresso è usato per indicare alla CPU che il dispositivo esterno indirizzato non è ancora pronto per il trasferimento di informazioni. In questo modo si riesce a sincronizzare la CPU con qualsiasi dispositivo esterno che operi ad una velocità diversa. L'ingresso è attivo basso.

$\overline{\text{RESET}}$: (Azzeramento); questo ingresso portato al livello basso, prepara l'inizio delle operazioni forzando la CPU nelle seguenti fasi:
1) azzera il Program Counter
2) azzera il f-f di abilitazione delle interruzioni
3) azzera i registri I ed R
4) programma la risposta alle interruzioni nel modo 0.

Infine i bus dei dati e degli indirizzi si pongono nello stato di alta impedenza per tutto il periodo di reset.

ESERCIZI PROPOSTI

1. Mediante l'utilizzo di una memoria PR0M da 256 × 4 bit, si vuole realizzare l'automazione delle seguenti funzioni:

$f_4 = A\,\overline{B}\,C\,D\,\overline{E}\,\overline{F}\,G\,\overline{H}$
$f_3 = B\,\overline{C}\,F\,\overline{G}$
$f_2 = A\,\overline{C}\,G$
$f_1 = A\,\overline{B}\,\overline{C}\,F + E\,\overline{F}\,G\,H + \overline{A}\,\overline{B}\,C\,\overline{D}\,E\,F\,\overline{G}\,\overline{H} + D\,\overline{E}\,G\,H + A\,C\,D\,\overline{F} + \overline{B}\,\overline{C}\,\overline{D}\,\overline{G} + \overline{A}\,C\,D\,H + B\,C\,D\,\overline{F}.$

Si indichi in quali dei 256 indirizzi, deve essere scritto un 1 per ciascuna funzione.

2. Si scriva la tabella della verità per la programmazione di una rete PLA che debba realizzare le seguenti funzioni:

$f_1 = A\,\overline{B}\,\overline{C}\,D + A\,\overline{C} + B\,C\,D + \overline{A}\,\overline{C}\,F$
$f_2 = \overline{D}\,E\,F + A\,\overline{B}\,D + A\,E\,F.$
$f_3 = \overline{A}\,\overline{B}\,C\,D + A\,\overline{B}\,C\,D\,E\,\overline{F}$
$f_4 = A\,\overline{C}\,\overline{D} + B\,C\,D + \overline{B}\,\overline{E}\,\overline{F}.$

3. Si disegni lo schema delle connessioni da realizzare in una PROM con parole di 8 bit, per realizzare un divisore di Moebius per 16.

4. Mediante l'uso di integrati 74 LS 365, si realizzi un bus bidirezionale a 6 bit di tipo tri-state, nel quale il flusso dei dati in un verso o nel verso opposto avvenga su comando di due segnali IN ed OUT.

5. Si scriva una tabella di flusso degli stati di uscita, per ottenere un divisore di frequenza per 5 che utilizzi una memoria PR0M. Si facciano le ipotesi che cambi una sola uscita alla volta e che si parta dalla condizione con le uscite tutte a livello alto.

CAPITOLO SECONDO

LA PROGRAMMAZIONE

1. Generalità

Che cosa si intende per programma? Un programma può essere definito come una sequenza ordinata di istruzioni che l'elaboratore è chiamato ad eseguire per svolgere il problema richiesto.

La risoluzione di un problema mediante l'uso di un computer può essere pensata suddivisa in quattro parti che svolgono ordinatamente nel tempo le seguenti funzioni:

- definizione o analisi del problema e scelta dell'algoritmo da utilizzare.
- stesura del diagramma di flusso o flow-chart che descrive l'algoritmo scelto
- codifica del programma
- messa a punto del programma

Analisi del problema

Durante le fasi di analisi devono essere definite in modo completo tutte le caratteristiche del problema da risolvere, quali: la definizione degli input-output necessari, i controlli di validità, la definizione dei tempi di risposta etc...

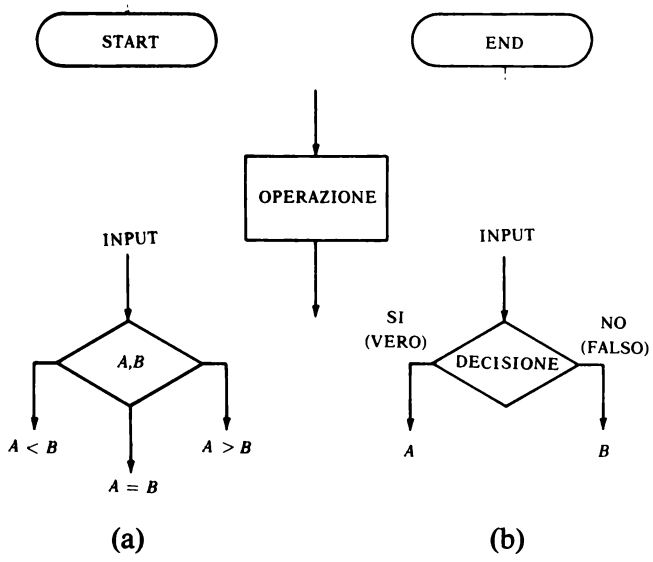
Deve inoltre essere scelto l'algoritmo che realizza la sequenza operativa richiesta dal problema tenendo conto di scegliere fra i possibili quelli che ottimizzano i seguenti punti:

- velocità di esecuzione
- dimensione della memoria richiesta
- minimizzazione del costo del programma.

Diagramma di flusso

Questa fase prevede la stesura del flow-chart, cioè di uno schema a blocchi che evidenzia l'intero procedimento e la sequenza operativa dell'algoritmo scelto.

Per la stesura dei flow-chart sono utilizzati dei simboli standard, fra i quali i più ricorrenti sono:

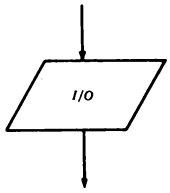


indicano il punto iniziale e finale di un programma o sottoprogramma

indica una generica istruzione che deve essere eseguita

all'interno del blocco decisionale viene indicata l'espressione logica (a) o il confronto (b) in base al cui valore si decide quale percorso deve essere seguito in uscita.

questo simbolo permette la connessione di una parte di flow-chart con un altro



indica una o più operazioni di input-output specificate all'interno del simbolo.

Il più elementare flow-chart è il seguente:

Se il gruppo di istruzioni, ad esempio, dovesse essere eseguito per n volte consecutive anziché una volta sola, il flow-chart corrispondente è quello di fig. 1 in cui compare una struttura ripetitiva (loop). Una volta che il registro in cui è caricato il numero delle volte che il gruppo di istruzioni deve essere eseguito, si è azzerato, il programma ha termine.

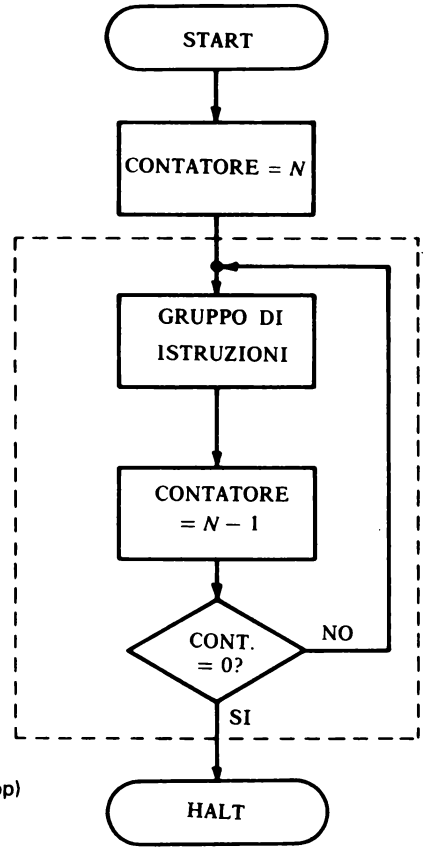


FIG. 1. -
Esempio di flow-chart con struttura ripetitiva (loop)

Codifica del programma

In questa fase si codificano le istruzioni nel linguaggio di programmazione scelto. Il programma viene poi trasferito su un qualsiasi supporto fisico (scheda, nastro perforato, minidisco, RAM etc.). Una volta caricato viene poi eseguito inserendo dati in input.

Messa a punto del programma

La messa a punto del programma o «debugging» consiste nell'insieme di test che debbono essere fatti per eliminare gli eventuali errori presenti nel programma codificato ed è necessaria per assicurare il corretto funzionamento del sistema.

2. Linguaggi di programmazione

Un calcolatore è in grado di comprendere delle istruzioni solo in termini di 0 ed 1 (linguaggio macchina) per cui l'operatore dovrebbe codificare i vari passi del programma esclusivamente in binario, ciò sarebbe estremamente laborioso oltre che soggetto a frequenti errori di trascrizione nel caso di stesura di programmi complessi. Per ovviare a questi inconvenienti sono stati introdotti dei «linguaggi di programmazione».

Un linguaggio di programmazione infatti rende possibile la scrittura di un programma in codice mnemonico, cioè in una forma che permette all'utente di specificare le varie operazioni e gli operandi con delle notazioni simboliche alfanumeriche anziché con notazioni binarie rendendo molto più agevole il compito del programmatore. Infatti anzichè scrivere 01111000 (istruzione di caricamento dell'accumulatore con il contenuto del registro B per lo Z 80)

è più semplice scrivere la stessa istruzione in codice esadecimale:

78

e lo è ancor di più in codice mnemonico o «assembly»:

LD A, B (load B in A)

I linguaggi di programmazione possono essere divisi per quanto riguarda i microprocessori in due categorie:

1. Linguaggi assembler o a basso livello
2. Linguaggi evoluti o ad alto livello

Nella prima categoria ad ogni espressione composta da una sequenza opportuna di caratteri detta «statement» corrisponde una sola istruzione binaria. Esiste cioè una corrispondenza 1:1 fra l'istruzione mnemonica e quella comprensibile all'elaboratore.

I linguaggi assembler sono orientati alla macchina in quanto presuppongono una conoscenza accurata dell'architettura interna da parte dell'operatore. Ciò consente una utilizzazione ottimale dell'apparato permettendo una maggiore velocità di esecuzione dei programmi. Rispetto al linguaggio macchina un linguaggio assembler offre i seguenti vantaggi:
- minore possibilità di errori di scrittura
- istruzioni più facili da ricordare
- errori di sintassi riconosciuti dall'assemblatore

Per contro c'è lo svantaggio di dover disporre di un dispositivo in più: l'assemblatore.

Nella seconda categoria invece ad ogni statement corrispondono più istruzioni binarie e di conseguenza la stesura del programma risulta molto più breve di quella ottenibile usando l'assembly (fig. 2).

```
        LD A, n1                    INPUT A, B
        LD B, n1                    LET E = A * B
        LD C, n2-1                  PRINT "PRODOTTO = ", E
P1      ADD A, B                    STOP
        DEC C
        JPNZ P1
        LDC, A
        OUT (C), C
        HALT
          (a)                         (b)
```

FIG. 2. - Prodotto di due numeri usando un linguaggio Assembly (a) oppure il linguaggio evoluto Basic (b).

Inoltre si ha una maggiore semplicità di programmazione in quanto i linguaggi evoluti non presuppongono da parte dell'operatore conoscenze specifiche del sistema utilizzato, come ad esempio: il numero dei registri, gli indirizzi di memoria etc.

I linguaggi ad alto livello sono quindi indipendenti dalla macchina e orientati verso il problema.

Esempi di linguaggi evoluti sono il Fortran (applicazioni scientifiche), Cobol (applicazioni commerciali), PL/1 (applicazioni commerciali-scientifiche), Basic, Pascal, etc.

Naturalmente sia che si abbia a che fare con un linguaggio Assembler che evoluto, occorre poter disporre di un programma specifico chiamato rispettivamente assemblatore o compilatore, fornito dal costruttore insieme al sistema. Tale programma permette di tradurre gli statement del programma *sorgente* (scritto dal programmatore) nel corrispondente programma *oggetto* in codice binario comprensibile alla macchina (fig. 3).

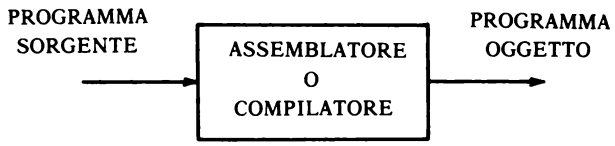


FIG. 3.

Linguaggio assembler o linguaggio evoluto?

La scelta di un linguaggio anziché di un altro dipende dal tipo di applicazione richiesta. In genere si può dire che un linguaggio assembler viene utilizzato:
- nei programmi di lunghezza piccolo-media
- nelle applicazioni in cui è importante contenere il costo della memoria (un assemblatore richiede molto meno memoria di un compilatore e quindi è a più basso costo rispetto a quest'ultimo)
- nelle applicazioni di controllo in tempo reale
- quando si programmano funzioni dipendenti dalla macchina (per esempio dispositivi di I/0 pilotati in interrupt), funzioni ove convenga risparmiare tempo etc.

Un linguaggio ad alto livello è invece usato:
- nei programmi di media-grande lunghezza
- nelle applicazioni che richiedono una grande capacità di memoria
- nei casi in cui sono richiesti più calcoli che non operazioni di input/output o di controllo.

Anche nel caso di sistemi utilizzanti microprocessori tutte le istruzioni per poter essere eseguite devono essere espresse in codice binario.

Le sempre maggiori capacità dei microprocessori richiedono ogni giorno programmi più complessi, nello stesso tempo i costi orari di programmazione aumentano costantemente assai rapidamente. Questi fattori hanno finora ostacolato l'adattamento dei linguaggi evoluti ai microprocessori. Infatti, benché tali linguaggi rendono la programmazione più semplice e più rapida rispetto ai linguaggi assembly, richiedono tuttavia maggiori tempi di esecuzione ed una memoria più ampia.

Nel caso del μ P Z 80 il passaggio dal programma sorgente a quello oggetto può essere definito mediante un compilatore Basic o di un assemblatore memorizzati su Rom o su cassetta magnetica (nel secondo caso), oppure direttamente dal programmatore per mezzo del set di istruzioni specifico necessario per tradurre gli statement in codice operativo. In quest'ultimo caso si parla di *assemblaggio manuale.* Nell'assemblaggio manuale occorre stabilire a priori la zona di memoria in cui deve essere caricato il programma.

La carta di riferimento dello Z 80 fornita dalla SGS oltre a riportare il codice operativo binario del set delle istruzioni, riporta anche il codice

esadecimale delle stesse. Questo perché normalmente nei sistemi utilizzanti lo Z 80 come CPU esiste un programma di conversione esadecimale-binario chiamato *loader* (fig. 4).

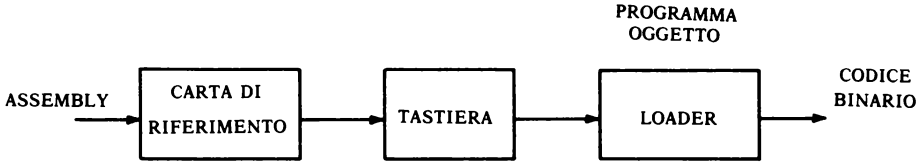


FIG. 4. - Assemblaggio manuale.

L'uso del codice esadecimale comporta una minore probabilità di commettere errori ed una più spedita ricerca degli stessi.

Infatti se

11011101 è l'istruzione corretta
e 11011011 è quella caricata

le stesse istruzioni in esadecimale sono:

DD
DB

L'errore commesso risulta quindi in fase di verifica molto più evidente al programmatore se l'istruzione è stata caricata in esadecimale anziché in codice binario.

3. Flag

Come si è già visto nel capitolo precedente esistono nello Z 80 sei flags (bit di condizione) che forniscono informazioni a secondo del loro stato in un dato istante (fig. 5).

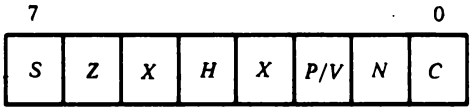


FIG. 5. - Registro di flag.

dove

C = flag di carry
P/V = flag di parity-overflow
Z = flag di zero
X = non usato
N = flag di add/subtract
H = flag di half-carry
S = flag di segno

Ognuno di questi bit contenuti nel registro di flag del microprocessore viene settato (= 1) o resettato (= 0) in base al risultato di una operazione precedente. È quindi possibile come si vedrà in seguito condizionare l'evoluzione del programma in base al risultato di operazioni svolte al suo interno testando tali bit di condizione.

Flag di carry

Se una istruzione produce un riporto (da addizionare) o un prestito (sottrazione o comparazione) dal bit più significativo il flag è settato (= 1), altrimenti resettato (= 0).

Flag di addizione-sottrazione

Dopo l'esecuzione di una addizione il flag è resettato mentre dopo una operazione di sottrazione è settato.

Flag di parità e overflow

Se il numero di 1 del risultato di una operazione è pari il flag è settato, altrimenti resettato. Nel caso di somma tra due numeri con lo stesso segno che danno come risultato un numero di segno differente (overflow) il flag è settato.

Flag di half-carry

Il flag è settato se l'istruzione aritmetica ha causato un riporto dal bit 3 al bit 4, altrimenti è resettato.

Flag di zero

Se il risultato di una operazione è nullo il bit è settato, in caso contrario resettato.

Flag di segno

Se il bit più significativo del risultato di una operazione vale 1 il flag è settato, altrimenti resettato.

4. Set di istruzioni del microprocessore Z 80

Come si è già visto nel capitolo precedente in ogni microcomputer le istruzioni sono codificate in una o più parole di lunghezza pari al parallelismo del microprocessore. Nel caso dello Z 80 una istruzione può essere composta fino a quattro parole. La prima parte della istruzione è il codice operativo e definisce l'operazione che la CPU deve eseguire.

La seconda parte definisce invece il campo operando sul quale si esegue l'operazione. Quest'ultima parte non è presente nel caso di istruzioni attinenti ai registri interni della CPU in quanto il loro indirizzo è già inglobato nel codice operativo (fig. 6).

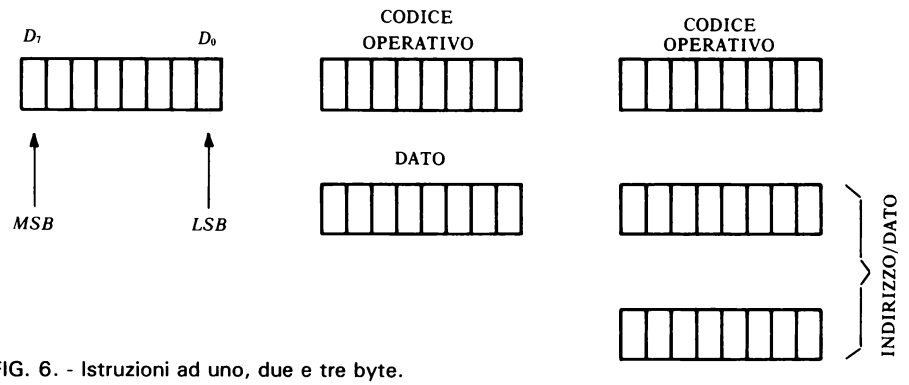


FIG. 6. - Istruzioni ad uno, due e tre byte.

Il μP Z 80 può eseguire 158 tipi diversi di istruzioni che possono essere divise nei seguenti gruppi principali:
- caricamento, scambio, trasferimento e ricerca dati
- aritmetiche e logiche
- rotazione e scorrimento
- set, reset e test di bit
- salto e subroutine
- input-output e di controllo (istruzioni dipendenti da decisioni prese nell'ambito del programma).

Nel presente paragrafo saranno introdotte alcune di queste istruzioni per la stesura di semplici programmi con l'intento di far comprendere l'utilizzo della carta di riferimento. Si lascerà poi all'abilità del programmatore il compito di utilizzare e scegliere convenientemente le altre a seconda del problema trattato.

SET DI ISTRUZIONI DELLO Z 80
(Dal manuale tecnico SGS)

Istruzione	C	Z	P/V	S	N	H	Commenti
ADD A, s; ADC A, s	↕	↕	V	↕	0	↕	Somma ad 8-bit o somma ad 8-bit con riporto
SUB s; SBC A, s, CP s, NEG	↕	↕	V	↕	1	↕	Sottrazione ad 8-bit, sottrazione con riporto, confronto e negazione dell'accumulatore
AND s	0	↕	P	↕	0	1	} Operazioni logiche
OR s; XOR s	0	↕	P	↕	0	0	
INC s	•	↕	V	↕	0	↕	Incremento ad 8-bit
DEC m	•	↕	V	↕	1	↕	Decremento ad 8-bit
ADD DD, ss	↕	•	•	•	0	X	Somma a 16-bit
ADC HL, ss	↕	↕	V	↕	0	X	Somma a 16-bit con riporto
SBC HL, ss	↕	↕	V	↕	1	X	Sottrazione a 16-bit con riporto
RLA; RLCA, RRA, RRCA	↕	•	•	•	0	0	Rotazione dell'accumulatore
RLm; RLCm; RRm; RRCm SLA m; SRA m; SRL m	↕	↕	P	↕	0	0	Rotazione e scorrimento di posizione di memoria
RLD, RRD	•	↕	P	↕	0	0	Rotazione di un digit a sinistra od a destra
DAA	↕	↕	P	↕	•	↕	Aggiustamento decimale dell'accumulatore
CPL	•	•	•	•	1	1	Complemento dell'accumulatore
SCF	1	•	•	•	0	0	Set del flag di riporto (carry)
CCF	↕	•	•	•	0	X	Complemento del flag di riporto (carry)
IN r, (C)	•	↕	P	↕	0	0	Input indiretto
INI; IND; OUTI; OUTD	•	↕	X	X	1	X	} Input od output di un blocco di dati
INIR; INDR; OTIR; OTDR	•	1	X	X	1	X	} Z = 0 se B ≠ 0, altrimenti Z = 1
LDI, LDD	•	X	↕	X	0	0	} Istruzioni di trasferimento di un blocco di dati
LDIR, LDDR	•	X	0	X	0	0	} P/V = 1 se BC ≠ 0, altrimenti P/V = 0
CPI, CPIR, CPD, CPDR	•	↕	↕	X	1	X	Istruzioni di ricerca in un blocco di dati Z = 1 se A = (HL), altrimenti Z = 0 P/V = 1 se BC ≠ 0, altrimenti P/V = 0
LD A, I; LD A, R	•	↕	IFF	↕	0	0	Lo stato del flip-flop di abilitazione delle interruzioni (IFF) è copiato nel flag P/V
BIT b, s	•	↕	X	X	0	1	Lo stato del bit b della posizione di memoria s è copiato nel flag z
NEG	↕	↕	V	↕	1	↕	Negazione dell'accumulatore

In questa tabella sono adottate le seguenti notazioni:

Simbolo	Operazione
C	Flag di riporto (carry). C = 1 se l'operazione ha generato un riporto dal bit più significativo dell'operando o del risultato.
Z	Flag di zero. Z = 1 se il risultato dell'operazione è zero.
S	Flag di segno. S = 1 se il bit più significativo del risultato è 1.
P/V	Flag di parità/traboccamento (parity/overflow). La parità (P) ed il traboccamento (V) utilizzano lo stesso flag. Le operazioni logiche modificano questo flag con la parità del risultato, mentre le operazioni aritmetiche modificano questo flag con l'eventuale overflow del risultato. Se P/V indica parità, P/V= 1 se il risultato è pari, P/V= 0 se il risultato è dispari. P/V= 1 se il risultato della operazione genera un overflow.
H	Riporto parziale (half carry). H = 1 se la somma o la sottrazione generano un riporto nel (od un riporto negativo dal) bit 4 dell'accumulatore.
N	Flag di somma/sottrazione. N = 1 se la precedente operazione era una sottrazione.
↕	Il flag è modificato in accordo con il risultato dell'operazione.
•	Il flag non viene modificato dall'operazione.
0	Il flag viene posto a 0 dall'operazione.
1	Il flag viene posto ad 1 dall'operazione.
X	Il flag viene posizionato casualmente.
V	Il flag P/V viene modificato in accordo con l'overflow risultante.
P	Il flag P/V viene modificato in accordo con la parità risultante.
r	Uno qualsiasi dei registri A, B, C, D, E, H, L della CPU.
s	Qualsiasi singola posizione di memoria (8-bit).
ss	Qualsiasi doppia posizione di memoria (16-bit).
ii	Uno dei due registri indice IX o IY.
R	Contatore di rinfresco.
n	Un numero ad 8-bit, cioè nel campo < 0, 255 >.
nn	Un numero a 16-bit, cioè nel campo < 0, 65535 >.
m	Qualsiasi posizione di memoria ad 8-bit.

Codice mnemonico	Operazione Simbolica	Flag						Codice OP			N. di Byte	N. di cicli M	N. di stati T	Commenti
		C	Z	P/V	S	N	H	76	543	210				
LD r, r'	r ← r'	•	•	•	•	•	•	01	r	r'	1	1	4	r, r' | Reg.
LD r, n	r ← n	•	•	•	•	•	•	00	r	110	2	2	7	000 | B
								←	n	→				001 | C
LD r, (HL)	r ← (HL)	•	•	•	•	•	•	01	r	110	1	2	7	010 | D
LD r, (IX+d)	r ← (IX+d)	•	•	•	•	•	•	11	011	101	3	5	19	011 | E
								01	r	110				100 | H
								←	d	→				101 | L
LD r, (IY+d)	r ← (IY+d)	•	•	•	•	•	•	11	111	101	3	5	19	111 | A
								01	r	110				
								←	d	→				
LD (HL), r	(HL) ← r	•	•	•	•	•	•	01	110	r	1	2	7	
LD (IX+d), r	(IX+d) ← r	•	•	•	•	•	•	11	011	101	3	5	19	
								01	110	r				
								←	d	→				
LD (IY+d), r	(IY+d) ← r	•	•	•	•	•	•	11	111	101	3	5	19	
								01	110	r				
								←	d	→				
LD (HL), n	(HL) ← n	•	•	•	•	•	•	00	110	110	2	3	10	
								←	n	→				
LD (IX+d), n	(IX+d) ← n	•	•	•	•	•	•	11	011	101	4	5	19	
								00	110	110				
								←	d	→				
								←	n	→				
LD (IY+d), n	(IY+d) ← n	•	•	•	•	•	•	11	111	101	4	5	19	
								00	110	110				
								←	d	→				
								←	n	→				
LD A, (BC)	A ← (BC)	•	•	•	•	•	•	00	001	010	1	2	7	
LD A, (DE)	A ← (DE)	•	•	•	•	•	•	00	011	010	1	2	7	
LD A, (nn)	A ← (nn)	•	•	•	•	•	•	00	111	010	3	4	13	
								←	n	→				
								←	n	→				
LD (BC), A	(BC) ← A	•	•	•	•	•	•	00	000	010	1	2	7	
LD (DE), A	(DE) ← A	•	•	•	•	•	•	00	010	010	1	2	7	
LD (nn), A	(nn) ← A	•	•	•	•	•	•	00	110	010	3	4	13	
								←	n	→				
								←	n	→				
LD A, I	A ← I	•	‡	IFF	‡	0	0	11	101	101	2	2	9	
								01	010	111				
LD A, R	A ← R	•	‡	IFF	‡	0	0	11	101	101	2	2	9	
								01	011	111				
LD I, A	I ← A	•	•	•	•	•	•	11	101	101	2	2	9	
								01	000	111				
LD R, A	R ← A	•	•	•	•	•	•	11	101	101	2	2	9	
								01	001	111				

Note: r, r' significano uno qualsiasi dei registri A, B, C, D, E, H, L.
IFF significa che il contenuto del flip-flop di abilitazione delle interruzioni (IFF) è copiato nel flag P/V.

Flag: • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
‡ = il flag viene modificato in accordo col risultato dell'operazione.

Codice mnemonico	Operazione Simbolica	Flag						Codice OP			N. di Byte	N. di cicli M	N. di stati T	Commenti
		C	Z	P/V	S	N	H	76	543	210				
LD dd, nn	dd ← nn	•	•	•	•	•	•	00 ← ←	dd0 n n	001 → →	3	3	10	dd: 00 BC, 01 DE, 10 HL, 11 SP
LD IX, nn	IX ← nn	•	•	•	•	•	•	11 00 ← ←	011 100 n n	101 001 → →	4	4	14	
LD IY, nn	IY ← nn	•	•	•	•	•	•	11 00 ← ←	111 100 n n	101 001 → →	4	4	14	
LD HL, (nn)	H ← (nn+1) L ← (nn)	•	•	•	•	•	•	00 ← ←	101 n n	010 → →	3	5	16	
LD dd, (nn)	dd_H ← (nn+1) dd_L ← (nn)	•	•	•	•	•	•	11 01 ← ←	101 dd1 n n	101 011 → →	4	6	20	
LD IX, (nn)	IX_H ← (nn+1) IX_L ← (nn)	•	•	•	•	•	•	11 00 ← ←	011 101 n n	101 010 → →	4	6	20	
LD IY, (nn)	IY_H ← (nn+1) IY_L ← (nn)	•	•	•	•	•	•	11 00 ← ←	111 101 n n	101 010 → →	4	6	20	
LD (nn), HL	(nn+1) ← H (nn) ← L	•	•	•	•	•	•	00 ← ←	100 n n	010 → →	3	5	16	
LD (nn), dd	(nn+1) ← dd_H (nn) ← dd_L	•	•	•	•	•	•	11 01 ← ←	101 dd0 n n	101 011 → →	4	6	20	
LD (nn), IX	(nn+1) ← IX_H (nn) ← IX_L	•	•	•	•	•	•	11 00 ← ←	011 100 n n	101 010 → →	4	6	20	
LD (nn), IY	(nn+1) ← IY_H (nn) ← IY_L	•	•	•	•	•	•	11 00 ← ←	111 100 n n	101 010 → →	4	6	20	
LD SP, HL	SP ← HL	•	•	•	•	•	•	11	111	001	1	1	6	
LD SP, IX	SP ← IX	•	•	•	•	•	•	11 11	011 111	101 001	2	2	10	
LD SP, IY	SP ← IY	•	•	•	•	•	•	11 11	111 111	101 001	2	2	10	
PUSH qq	(SP-2) ← qq_L (SP-1) ← qq_H	•	•	•	•	•	•	11	qq0	101	1	3	11	qq: 00 BC, 01 DE, 10 HL, 11 AF
PUSH IX	(SP-2) ← IX_L (SP-1) ← IX_H	•	•	•	•	•	•	11 11	011 100	101 101	2	4	15	
PUSH IY	(SP-2) ← IY_L (SP-1) ← IY_H	•	•	•	•	•	•	11 11	111 100	101 101	2	4	15	
POP qq	qq_H ← (SP+1) qq_L ← (SP)	•	•	•	•	•	•	11	qq0	001	1	3	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP)	•	•	•	•	•	•	11 11	011 100	101 001	2	4	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP)	•	•	•	•	•	•	11 11	111 100	101 001	2	4	14	

Note: dd significa uno qualsiasi delle coppie di registri BC, DE, HL, SP
qq significa uno qualsiasi delle coppie di registri AF, BC, DE, HL
$(COPPIA)_H$, $(COPPIA)_L$ indicano il byte più significativo od il byte meno significativo della coppia indicata.
per esempio: BC_L = C, AF_H = A

Flag: • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
‡ = il flag viene modificato in accordo col risultato dell'operazione.

Codice mnemonico	Operazione Simbolica	Flag						Codice OP			N. di Byte	N. di cicli M	N. di stati T	Commenti
		C	Z	P/V	S	N	H	76	543	210				
EX DE, HL	DE ↔ HL	•	•	•	•	•	•	11	101	011	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	•	•	•	•	00	001	000	1	1	4	
EXX	$\begin{pmatrix}BC\\DE\\HL\end{pmatrix} \leftrightarrow \begin{pmatrix}BC'\\DE'\\HL'\end{pmatrix}$	•	•	•	•	•	•	11	011	001	1	1	4	Scambio dei due banchi di registri
EX (SP), HL	H ↔ (SP+1) L ↔ (SP)	•	•	•	•	•	•	11	100	011	1	5	19	
EX (SP), IX	IX_H ↔ (SP+1) IX_L ↔ (SP)	•	•	•	•	•	•	11 11	011 100	101 011	2	6	23	
EX (SP), IY	IY_H ↔ (SP+1) IY_L ↔ (SP)	•	•	•	•	•	•	11 11	111 100	101 011	2	6	23	
LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC - 1	•	•	① ↕	•	0	0	11 10	101 100	101 000	2	4	16	Load di (HL) in (DE), incremento dei puntatori e decremento del contatore di byte (BC)
LDIR	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1 Ripete finchè BC = 0	•	•	0	•	0	0	11 10	101 110	101 000	2 2	5 4	21 16	Se BC ≠ 0 Se BC = 0
LDD	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	•	•	① ↕	•	0	0	11 10	101 101	101 000	2	4	16	
LDDR	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1 Ripete finchè BC = 0	•	•	0	•	0	0	11 10	101 111	101 000	2 2	5 4	21 16	Se BC ≠ 0 Se BC = 0
CPI	A - (HL) HL ← HL+1 BC ← BC-1	•	② ↕	① ↕	↕	1	↕	11 10	101 100	101 001	2	4	16	
CPIR	A - (HL) HL ← HL+1 BC ← BC-1 Ripete finchè A = (HL) o BC = 0	•	② ↕	① ↕	↕	1	↕	11 10	101 110	101 001	2 2	5 4	21 16	Se BC=0 e A≠(HL) Se BC=0 e A= (HL)
CPD	A - (HL) HL ← HL-1 BC ← BC-1	•	② ↕	① ↕	↕	1	↕	11 10	101 101	101 001	2	4	16	
CPDR	A - (HL) HL ← HL-1 BC ← BC-1 Ripete finchè A = (HL) o BC = 0	•	② ↕	① ↕	↕	1	↕	11 10	101 111	101 001	2 2	5 4	21 16	Se BC=0 e A≠(HL) Se BC=0 e A= (HL)

Note: ① P/V è 0 se il risultato di BC-1 = 0, altrimenti P/V = 1
② Il Flag Z è 1 se A = (HL), altrimenti Z = 0

Flag: • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
↕ = il flag viene modificato in accordo col risultato dell'operazione.

Codice mnemonico	Operazione Simbolica	Flag						Codice OP			N. di Byte	N. di cicli M	N. di stati T	Commenti
		C	Z	P/V	S	N	H	76	543	210				
ADD A, r	A ← A + r	‡	‡	V	‡	0	‡	10	000	r	1	1	4	r | Reg. 000 | B 001 | C 010 | D 011 | E 100 | H 101 | L 111 | A
ADD A, n	A ← A + n	‡	‡	V	‡	0	‡	11	000	110	2	2	7	
								←	n	→				
ADD A, (HL)	A ← A + (HL)	‡	‡	V	‡	0	‡	10	000	110	1	2	7	
ADD A, (IX+d)	A ← A+(IX+d)	‡	‡	V	‡	0	‡	11	011	101	3	5	19	
								10	000	110				
								←	d	→				
ADD A, (IY+d)	A ← A + (IY+d)	‡	‡	V	‡	0	‡	11	111	101	3	5	19	
								10	000	110				
								←	d	→				
ADC A, s	A ← A+s+CY	‡	‡	V	‡	0	‡		001					s può essere r, n, (HL),(IX+d),(IY+d) come è mostrato per l'istruzione ADD
SUB s	A ← A - s	‡	‡	V	‡	1	‡		010					
SBC A, s	A ← A-s-CY	‡	‡	V	‡	1	‡		011					
AND s	A ← A ∧ s	0	‡	P	‡	0	1		100					I bit indicati sostituiscono 000 indicato per l'istruzione ADD.
OR s	A ← A ∨ s	0	‡	P	‡	0	0		110					
XOR s	A ← A ⊕ s	0	‡	P	‡	0	0		101					
CP s	A - s	‡	‡	V	‡	1	‡		111					
INC r	r ← r + 1	•	‡	V	‡	0	‡	00	r	100	1	1	4	
INC (HL)	(HL) ← (HL) + 1	•	‡	V	‡	0	‡	00	110	100	1	3	11	
INC (IX+d)	(IX+d) ←	•	‡	V	‡	0	‡	11	011	101	3	6	23	
	(IX+d) + 1							00	110	100				
								←	d	→				
INC (IY+d)	(IY+d) ←	•	‡	V	‡	0	‡	11	111	101	3	6	23	
	(IY+d) + 1							00	110	100				
								←	d	→				
DEC m	m ← m - 1	•	‡	V	‡	1	‡			101				m può essere r, (HL), (IX+d), (IY+d), come è mostrato per l'istruzione INC. Identici stati e formato dell'istruzione INC. Sostituire 100 con 101 nel codice OP.

Note: Il simbolo V nella colonna del flag P/V indica che il flag P/V contiene l'overflow del risultato dell'operazione. In modo simile il simbolo P indica parità. V = 1 significa overflow, V = 0 significa mancanza di overflow. P = 1 significa che la parità del risultato è pari, P = 0 significa che essa è dispari.

Flag: • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
‡ = il flag viene modificato in accordo col risultato dell'operazione.

Codice mnemonico	Operazione Simbolica	Flag						Codice OP			N. di Byte	N. di cicli M	N. di stati T	Commenti
		C	Z	P/V	S	N	H	76	543	210				
ADD HL, ss	HL ← HL + ss	‡	•	•	•	0	X	00	ss1	001	1	3	11	ss | Reg. 00 | BC 01 | DE 10 | HL 11 | SP
ADC HL, ss	HL ← HL+ss+CY	‡	‡	V	‡	0	X	11	101	101	2	4	15	
								01	ss1	010				
SBC HL, ss	HL ← HL-ss-CY	‡	‡	V	‡	1	X	11	101	101	2	4	15	
								01	ss0	010				
ADD IX, pp	IX ← IX + pp	‡	•	•	•	0	X	11	011	101	2	4	15	pp | Reg. 00 | BC 01 | DE 10 | IX 11 | SP
								00	pp1	001				
ADD IY, rr	IY ← IY + rr	‡	•	•	•	0	X	11	111	101	2	4	15	rr | Reg. 00 | BC 01 | DE 10 | IY 11 | SP
								00	rr1	001				
INC ss	ss ← ss + 1	•	•	•	•	•	•	00	ssU	011	1	1	6	
INC IX	IX ← IX + 1	•	•	•	•	•	•	11	011	101	2	2	10	
								00	100	011				
INC IY	IY ← IY + 1	•	•	•	•	•	•	11	111	101	2	2	10	
								00	100	011				
DEC ss	ss ← ss - 1	•	•	•	•	•	•	00	ss1	011	1	1	6	
DEC IX	IX ← IX - 1	•	•	•	•	•	•	11	011	101	2	2	10	
								00	101	011				
DEC IY	IY ← IY - 1	•	•	•	•	•	•	11	111	101	2	2	10	
								00	101	011				

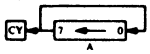
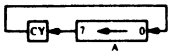
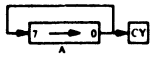
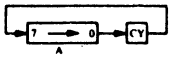
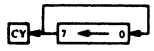
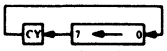
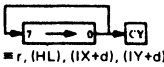
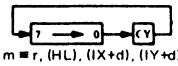
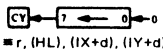
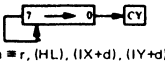
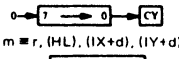
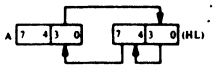
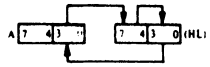
Note: ss è uno qualsiasi della coppia di registri BC, DE, HL, SP.
pp è uno qualsiasi della coppia di registri BC, DE, IX, SP.
rr è uno qualsiasi della coppia di registri BC, DE, IY, SP.

Flag : • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
‡ = il flag viene modificato in accordo col risultato dell'operazione.

Codice mnemonico	Operazione Simbolica	Flag						Codice OP			N. di Byte	N. di cicli M	N. di stati T	Commenti
		C	Z	P/V	S	N	H	76	543	210				
DAA	Converte il contenuto dell'Accumulatore in BCD dopo somma o sottrazione con operandi BCD	‡	‡	P	‡	•	‡	00	100	111	1	1	4	Aggiustamento decimale dell'accumulatore.
CPL	$A \leftarrow \overline{A}$	•	•	•	•	1	1	00	101	111	1	1	4	Complemento dell'accumulatore (complemento ad uno)
NEG	A ← 0 - A	‡	‡	V	‡	1	‡	11	101	101	2	2	8	Negazione dell'accumulatore (complemento a due)
								01	000	100				
CCF	$CY \leftarrow \overline{CY}$	‡	•	•	•	0	X	00	111	111	1	1	4	Complemento del flag di riporto.
SCF	CY ← 1	1	•	•	•	0	0	00	110	111	1	1	4	Set del flag di riporto.
NOP	Nessuna oper.	•	•	•	•	•	•	00	000	000	1	1	4	
HALT	Alt. della CPU	•	•	•	•	•	•	01	110	110	1	1	4	
DI	IFF ← 0	•	•	•	•	•	•	11	110	011	1	1	4	
EI	IFF ← 1	•	•	•	•	•	•	11	111	011	1	1	4	
IM 0	Seleziona il modo 0 di inter.	•	•	•	•	•	•	11	101	101	2	2	8	
								01	000	110				
IM 1	Seleziona il modo 1 di inter.	•	•	•	•	•	•	11	101	101	2	2	8	
								01	010	110				
IM 2	Selezione il modo 2 di inter.	•	•	•	•	•	•	11	101	101	2	2	8	
								01	011	110				

Note: IFF indica il flip-flop di abilitazione delle interruzioni.
CY indica il flip-flop di riporto (carry).

Flag : • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
‡ = il flag viene modificato in accordo col risultato dell'operazione.

Codice Mnemonico	Operazione Simbolica	Flag C	Z	P/V	S	N	H	Codice OP 76	543	210	N. di Byte	N. di cicli M	N. di stati T	Commenti
RLCA		‡	•	•	•	0	0	00	000	111	1	1	4	Rotazione circolare a sinistra dell'accumulatore
RLA		‡	•	•	•	0	0	00	010	111	1	1	4	Rotazione a sinistra dell'accumulatore
RRCA		‡	•	•	•	0	0	00	001	111	1	1	4	Rotazione circolare a destra dell'accumulatore
RRA		‡	•	•	•	0	0	00	011	111	1	1	4	Rotazione a destra dell'accumulatore
RLC r		‡	‡	P	‡	0	0	11 00	001 000	011 r	2	2	8	Rotazione circolare a sinistra del registro r
RLC (HL)		‡	‡	P	‡	0	0	11 00	001 000	011 110	2	4	15	r | Reg. 000 | B 001 | C 010 | D 011 | E 100 | H 101 | L 111 | A
RLC (IX+d)	 r, (HL), (IX+d), (IY+d)	‡	‡	P	‡	0	0	11 11 ← 00	011 001 d 000	101 011 → 110	4	6	23	
RLC (IY+d)		‡	‡	P	‡	0	0	11 11 ← 00	111 001 d 000	101 011 → 110	4	6	23	
RL m	 m ≡ r, (HL), (IX+d), (IY+d)	‡	‡	P	‡	0	0		010					Il formato dell'istruzione e gli stati sono eguali a quelli di RLC, m. Per formare il nuovo codice operativo sostituire 000 di RLC, m con il codice indicato.
RRC m	 m ≡ r, (HL), (IX+d), (IY+d)	‡	‡	P	‡	0	0		001					
RR m	 m ≡ r, (HL), (IX+d), (IY+d)	‡	‡	P	‡	0	0		011					
SLA m	 m ≡ r, (HL), (IX+d), (IY+d)	‡	‡	P	‡	0	0		100					
SRA m	 m ≡ r, (HL), (IX+d), (IY+d)	‡	‡	P	‡	0	0		101					
SRL m	 m ≡ r, (HL), (IX+d), (IY+d)	‡	‡	P	‡	0	0		111					
RLD		•	‡	P	‡	0	0	11 01	101 101	101 111	2	5	18	Rotazione di un digit a sinistra ed a destra attraverso l'accumulatore e la posizione (HL). Il contenuto della metà superiore dell'accumulatore non è modificato.
RRD		•	‡	P	‡	0	0	11 01	101 100	101 111	2	5	18	

Flag: • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
‡ = il flag viene modificato in accordo col risultato dell'operazione.

Codice mnemonico	Operazione Simbolica	Flag						Codice OP			N. di Byte	N. di cicli M	N. di stati T	Commenti
		C	Z	P/V	S	N	H	76	543	210				
BIT b, r	$Z \leftarrow \overline{r}_b$	•	‡	X	X	0	1	11	001	011	2	2	8	r | Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A b b | Bit Testato 000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
								01	b	r				
BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	•	‡	X	X	0	1	11	001	011	2	3	12	
								01	b	110				
BIT b, (IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	•	‡	X	X	0	1	11	011	101	4	5	20	
								11	001	011				
								←	d	→				
								01	b	110				
BIT b, (IY+d)	$Z \leftarrow \overline{(IY+d)}_b$	•	‡	X	X	0	1	11	111	101	4	5	20	
								11	001	011				
								←	d	→				
								01	b	110				
SET b, r	$r_b \leftarrow 1$	•	•	•	•	•	•	11	001	011	2	2	8	
								[11]	b	r				
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	•	•	•	•	11	001	011	2	4	15	
								[11]	b	110				
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	•	•	•	•	11	011	101	4	6	23	
								11	001	011				
								←	d	→				
								[11]	b	110				
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	•	•	•	•	11	111	101	4	6	23	
								11	001	011				
								←	d	→				
								[11]	b	110				
RES b, m	$s_b \leftarrow 0$ $m \equiv r, (HL), (IX+d), (IY+d)$							[10]						Per formare il nuovo codice OP, sostituire [11] di SET b, m con [10]. Flag e stati eguali alla istruzione di SET.

Note: La notazione s_b indica il bit b (da 0 a 7) della posizione di memoria s.

Flag : • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
‡ = il flag viene modificato in accordo col risultato dell'operazione.

Codice mnemonico	Operazione Simbolica	Flag						Codice OP			N. di Byte	N. di cicli M	N. di stati T	Commenti
		C	Z	P/V	S	N	H	76	543	210				
JP nn	PC ← nn	•	•	•	•	•	•	11	000	011	3	3	10	
								←	n	→				
								←	n	→				
JP cc, nn	Se la condizione cc è vera PC ← nn, altrimenti continua	•	•	•	•	•	•	11	cc	010	3	3	10	
								←	n	→				
								←	n	→				
JR e	PC ← PC + e	•	•	•	•	•	•	00	011	000	2	3	12	
								←	e-2	→				
JR C, e	Se C = 0, continua	•	•	•	•	•	•	00	111	000	2	2	7	Se la condizione non è assolta
								←	e-2	→				
	Se C = 1, PC ← PC + e										2	3	12	Se la condizione è assolta
JR NC, e	Se C = 1, continua	•	•	•	•	•	•	00	110	000	2	2	7	Se la condizione non è assolta
								←	e-2	→				
	Se C = 0, PC ← PC + e										2	3	12	Se la condizione è assolta
JR Z, e	Se Z = 0 continua	•	•	•	•	•	•	00	101	000	2	2	7	Se la condizione non è assolta
								←	e-2	→				
	Se Z = 1, PC ← PC + e										2	3	12	Se la condizione è assolta
JR NZ, e	Se Z = 1, continua	•	•	•	•	•	•	00	100	000	2	2	7	Se la condizione non è assolta
								←	e-2	→				
	Se Z = 0 PC ← PC + e										2	3	12	Se la condizione è assolta
JP (HL)	PC ← HL	•	•	•	•	•	•	11	101	001	1	1	4	
JP (IX)	PC ← IX	•	•	•	•	•	•	11	011	101	2	2	8	
								11	101	001				
JP (IY)	PC ← IY	•	•	•	•	•	•	11	111	101	2	2	8	
								11	101	001				
DJNZ, e	B ← B-1 Se B = 0, continua	•	•	•	•	•	•	00	010	000	2	2	8	Se B = 0
								←	e-2	→				
	Se B ≠ 0, PC ← PC + e										2	3	13	Se B ≠ 0

cc	Condizione
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO par. odd
101	PE par. even
110	P sign posit.
111	M sign negat.

Note: e rappresenta l'estensione nel modo di indirizzamento relativo.
e è un numero con segno, complemento a due compreso nel valore < -126, 129 >.
e-2 nel codice operativo fornisce un indirizzo reale PC + e, poiché il Program Counter viene incrementato di 2 prima della somma con e.

Flag: • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
‡ = il flag viene modificato in accordo col risultato dell'operazione.

Codice mnemonico	Operazione Simbolica	Flag						Codice OP			N. di Byte	N. di cicli M	N. di stati T	Commenti
		C	Z	P/V	S	N	H	76	543	210				
CALL nn	(SP-1) ←PC_H	•	•	•	•	•	•	11	001	101	3	5	17	
	(SP-2) ←PC_L							←	n	→				
	PC ←nn							←	n	→				
CALL cc, nn	Se la condizione cc è falsa continua, altrimenti come CALL nn	•	•	•	•	•	•	11	cc	100	3	3	10	Se cc è falso
								←	n	→				
								←	n	→	3	5	17	Se cc è vero
RET	PC_L ← (SP)	•	•	•	•	•	•	11	001	001	1	3	10	
	PC_H ← (SP+1)													
RET cc	Se la condizione cc è falsa continua, altrimenti come RET	•	•	•	•	•	•	11	cc	000	1	1	5	Se cc è falso
											1	3	11	Se cc è vero
RETI	Ritorno dalla interruzione	•	•	•	•	•	•	11	101	101	2	4	14	
								01	001	101				
RETN	Ritorno dalla interruzione non mascherabile	•	•	•	•	•	•	11	101	101	2	4	14	
								01	000	101				
RST p	(SP-1) ←PC_H	•	•	•	•	•	•	11	t	111	1	3	11	
	(SP-2) ←PC_L													
	PC_H ←0													
	PC_L ←P													

cc	Condizione
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO par. odd
101	PE par. even
110	P sign posit.
111	M sign negat.

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Flag : • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
↕ = il flag viene modificato in accordo col risultato dell'operazione.

Codice mnemonico	Operazione Simbolica	Flag						Codice OP			N. di Byte	N. di cicli M	N. di stati T	Commenti
		C	Z	P/V	S	N	H	76	543	210				
IN A, (n)	A ← (n)	•	•	•	•	•	•	11	011	011	2	3	11	n su $A_0 \sim A_7$
								←	n	→				Acc su $A_8 \sim A_{15}$
IN r, (C)	r ← (C) se r = 110 solamente i flag saranno modificati	•	↕ ①	P	↕	0	↕	11	101	101	2	3	12	C su $A_0 \sim A_7$
								01	r	000				B su $A_8 \sim A_{15}$
INI	(HL) ← (C)	•	↕	X	X	1	X	11	101	101	2	4	16	C su $A_0 \sim A_7$
	B ← B - 1							10	100	010				B su $A_8 \sim A_{15}$
	HL ← HL + 1													
INIR	(HL) ← (C)	•	1	X	X	1	X	11	101	101	2	5	21	C su $A_0 \sim A_7$
	B ← B - 1							10	110	010		(Se B≠0)		B su $A_8 \sim A_{15}$
	HL ← HL + 1										2	4	16	
	Ripete finchè B = 0		①									(Se B= 0)		
IND	(HL) ← (C)	•	↕	X	X	1	X	11	101	101	2	4	16	C su $A_0 \sim A_7$
	B ← B - 1							10	101	010				B su $A_8 \sim A_{15}$
	HL ← HL-1													
INDR	(HL) ← (C)	•	1	X	X	1	X	11	101	101	2	5	21	C su $A_0 \sim A_7$
	B ← B - 1							10	111	010		(Se B≠0)		B su $A_8 \sim A_{15}$
	HL ← HL-1										2	4	16	
	Ripete finchè B = 0											(Se B= 0)		
OUT (n), A	(n) ← A	•	•	•	•	•	•	11	010	011	2	3	11	n su $A_0 \sim A_7$
								←	n	→				Acc su $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	•	•	•	•	11	101	101	2	3	12	C su $A_0 \sim A_7$
			①					01	r	001				B su $A_8 \sim A_{15}$
OUTI	(C) ← (HL)	•	↕	X	X	1	X	11	101	101	2	4	16	C su $A_0 \sim A_7$
	B ← B - 1							10	100	011				B su $A_8 \sim A_{15}$
	HL ← HL+1													
OTIR	(C) ← (HL)	•	1	X	X	1	X	11	101	101	2	5	21	C su $A_0 \sim A_7$
	B ← B - 1							10	110	011		(Se B≠0)		B su $A_8 \sim A_{15}$
	HL ← HL+1													
	Ripete finchè B = 0		①								2	4 (Se B=0)	16	
OUTD	(C) ← (HL)	•	↕	X	X	1	X	11	101	101	2	4	16	C su $A_0 \sim A_7$
	B ← B - 1							10	101	011				B su $A_8 \sim A_{15}$
	HL ← HL-1													
OTDR	(C) ← (HL)	•	1	X	X	1	X	11	101	101	2	5	21	C su $A_0 \sim A_7$
	B ← B-1							10	111	011		(Se B≠ 0)		B su $A_8 \sim A_{15}$
	HL ← HL-1													
	Ripete finchè B = 0										2	4 (Se B= 0)	16	

Note: ① Se il risultato di B-1 è zero il flag Z viene posto ad uno, altrimenti è posto a zero.

Flag : • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
↕ = il flag viene modificato in accordo col risultato dell'operazione.

Utilizzo della carta di riferimento per la stesura di semplici programmi da eseguire tramite un microcomputer utilizzante come unità centrale di processo lo Z 80

Ogni programma può essere pensato costituito dalle seguenti parti:
— un flow-chart se la logica del programma è complessa
— l'indirizzo di memoria a partire dal quale viene caricato
— la lista in codice mnemonico (assembly)
— la lista in codice oggetto
— note di commento

Caricamento del contenuto del registro B nel registro C

Il flow-chart è il seguente:

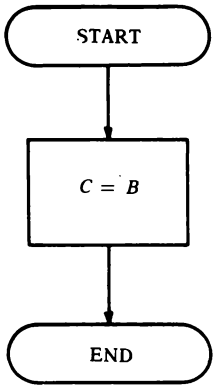


FIG. 7.

L'istruzione che esegue il caricamento è la prima del set di istruzioni (pag. 41).

Mnemonic	Symbolic operation	Flags	Operative code	N° of cycles	Comments	
					r r'	Registers
LD r,r'	r←r'	CZP/VSNH	01 r r'	4	0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 1	B C D E H L A

Questa istruzione carica il contenuto di r' in r, con r' ed r due tra i registri ad otto bit della CPU. Il puntino in corrispondenza dei flags informa l'operatore che essi non variano il contenuto dopo questa istruzione.

Il codice operativo (tenendo conto della tabella riportata nei commenti) vale:

$$0\ 1\ r\ r' = (01001000)_2 = (4\ 8)_H$$

con H abbreviazione di esadecimale.

Il tempo di esecuzione dell'istruzione vale:

$$n \cdot \frac{1}{2{,}5 \cdot 10^6} = 4 \cdot 0{,}4 \cdot 10^{-6} = 1{,}6\ \mu s$$

con n numero di periodi di clock necessari e assumendo (cap. 10) pari a 2,5 MHz la frequenza del clock.

Il programma, caricato ad esempio a partire dalla locazione di memoria $(0100)_H$ con 01 parte alta dell'indirizzo (HI) e 00 parte bassa (L0), è il seguente:

Locazione di memoria	codice esadecimale oggetto	codice sorgente (assembly)	commenti
0100	48	LD C,B	carica il contenuto di B in C
0101	76	HALT	fine del programma

Le locazioni di memoria interessate, ognuna contenente un byte, sono quelle di indirizzo 0100 e 0101.

Esecuzione del programma:
dopo aver memorizzato le istruzioni e caricato l'indirizzo di partenza nel program-counter (registro specializzato a 16 bit) si dà il via all'esecuzione del programma, programma che viene eseguito in un tempo pari a:

	$4 \cdot 0{,}4\ \mu s$	per l'istruzione LD C,B
+		
	$4 \cdot 0{,}4\ \mu s$	per l'istruzione HALT (dalla carta di riferimento)
	$= 3{,}2\ \mu s$	

Esecuzione della somma di due numeri esadecimali con risultato ≦ FF

Flow-chart:

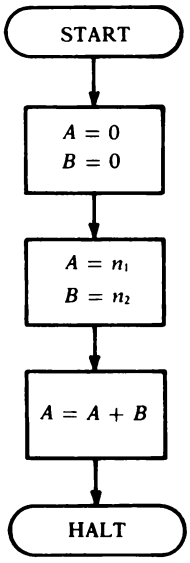


FIG. 8.

In questo esempio si incontrano due nuove istruzioni:
LD r,n ed ADD r.

La prima istruzione carica l'intero n a otto bit (r ← n) in uno qualsiasi dei registri, con r = A,B,C,D,E,H,L, lasciando inalterati i flags in un tempo pari a 7·0,4 μs.

Il codice oggetto, a due byte, vale:

$$\begin{array}{cccc} 00\,r\,110 & = & 00000110 & = 06 \\ \leftarrow\ n\ \rightarrow & & \leftarrow\ n\ \rightarrow & \end{array}$$

La seconda istruzione, aritmetica, ADD r somma il contenuto del registro r al contenuto dell'accumulatore (A = A + r) memorizzando il risultato nell'accumulatore. Il tempo di esecuzione è uguale a:

$$4\cdot 0{,}4\ \mu s$$

il codice operativo è:

$$10000r = 10000000 = 80$$

(nel caso del registro B)

Lo stato dei flags è il seguente:

C	Z	P/V	S	N	H
↕	↕	V	↕	O	↕

dove le freccette stanno ad indicare i flags che variano assumendo il valore 1 o 0 a seconda del risultato dell'operazione.

Il programma, caricato a partire dall'indirizzo 0102, risulta:

Locazione di memoria	Codice oggetto	Assembly	Commenti
0102	3E 00	LD A,00	azzerra l'accumulatore
0104	06 00	LD B,00	azzera B
0106	3E n_1	LD A,n_1	carica n_1 in A
0108	06 n_2	LD B, n_2	carica n_2 in B
010A	80	ADD A, B	somma B con A e poni il risultato in A
010B	76	HALT	

Una volta eseguito il programma l'accumulatore conterrà la somma dei numeri esadecimali n_1 ed n_2.

Il tempo di esecuzione, come può essere facilmente verificato, è uguale a 15,6 μs.

Rappresentazione binaria complemento a due

Una sequenza binaria di otto bit consente la rappresentazione dei numeri decimali da 0 a 255.

La rappresentazione binaria complemento a due con otto bit (fig. 9)

numero decimale	rappresentazione complemento a due con otto bit
+ 127	01111111 = 7F
+ 126	01111110 = 7E
+ 125	01111101 = 7D
.	.
.	.
.	.
0	00000000 = 00
— 1	11111111 = FF
— 2	11111110 = FE
.	.
.	.
.	.
— 128	10000000 = 80

FIG. 9.

consente invece la rappresentazione dei numeri positivi compresi tra 0 e + 127 in normale forma binaria e quelli negativi fino a — 128 effettuando il complemento a due (Vol. Elet. Dig. Cap. 1° parag. 5).

I numeri positivi hanno così sempre il bit più significativo uguale a zero, mentre quelli negativi uguale ad uno. Testando quindi solo il bit più significativo si è in grado di stabilire se il numero è positivo oppure negativo.

I seguenti esempi serviranno da chiarimento.

1. Rappresentare in binario complemento a due per otto bit i decimali +6, +15, -7, -32.

+ 6 = 00000110 = 06; + 15 = 00001111 = OF; -7 = 11111000 + 00000001 = 11111001 = F9; -32 = 11011111 (complemento di 32) + 00000001 = 11100000 = E0

2. Trovare i corrispondenti numeri decimali dei seguenti numeri complemento a due su otto bit: 00111111, 01000000, 10000100, 11000011.

00111111 = + 63; 01000000 = +64; 10000100 = 01111100 (complemento a due) = -124; 11000011 = 00111101 = -61.

Nel caso di una somma tra due numeri il cui risultato supera il valore + 127 oppure -128 si ha l'overflow o traboccamento. La condizione di overflow può essere segnalata controllando il bit più significativo di ogni addendo e della somma, in quanto si ha overflow tutte le volte che dalla somma di due numeri negativi si ottiene come risultato un numero positivo, oppure quando dalla somma di due numeri positivi si ottiene un numero negativo:

```
  10000000 (—128)          01111110 (+126)
+ 11111110 (—  2)        + 00010000 (+ 16)
------------------       ------------------
  01111110 (+126)          10001110 (—114)
```

Loop (anello)

Per loop si intende una sequenza di istruzioni che viene eseguita in modo ripetitivo sino a quando non si verifica una condizione di fine. In fig. 10 è riportato il flow-chart di un loop utilizzato per azzerare un registro in cui è caricato un certo valore esadecimale.

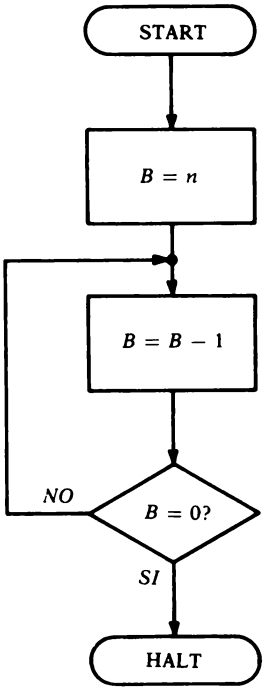


FIG. 10.

La prima istruzione del programma è una istruzione di caricamento con n ≦ FF). La seconda è una istruzione di decremento di un registro ad otto bit (DEC s) il cui codice operativo è il seguente:

00 r 101 = 00000101 = 05 (nel caso del registro B)

I bit di condizione sono tutti alterati da questa istruzione ad eccezione di quello di carry.

In particolare il flag di zero è messo ad 1 se il risultato è zero, messo a zero in caso contrario. Di conseguenza il flag Z varrà 1 soltanto nel caso che il contenuto del registro B è uguale a zero. La terza istruzione è una istruzione di salto condizionato JPNZ (salta se il flag Z è zero, ovvero il risultato non è zero), che viene eseguita fin tanto che il flag di zero vale 1. Una volta che Z viene portato a 0 il programma ha termine.

L'istruzione di salto condizionato, che non altera il registro dei flags, è una istruzione a tre byte. Il primo byte contiene il codice operativo, il secondo la parte bassa dell'indirizzo a cui si salta, il terzo la parte alta dell'indirizzo. Il codice mnemonico è JP cc,nn (salta a nn se la condizione cc è vera), nel presente caso (pag. 48) cc = NZ (non zero), cosicché il codice operativo risulta:

11000011	= C2	
n	n	parte bassa dell'indirizzo
n	n	parte alta dell'indirizzo

Il programma è il seguente:

Locazione di memoria	Codice oggetto	Assembly	Commenti
010C	06 n	LD B,n	carica n in B
P_1 010E	05	DEC B	decrementa B
010F	C2 0E 01	JPNZ P_1	Salta a P_1 se il flag Z è zero (contenuto di B $\neq$ 0)
0113	76	HALT	

Il tempo di esecuzione del programma, che dipende dal valore di n caricato nel registro B, è pari a:

$$\begin{aligned} & 7 \cdot 0{,}4\ \mu s \text{ (istruzione di caricamento)} \\ + \ & n \cdot 4 \cdot 0{,}4\ \mu s \text{ (istruzione di decremento)} \\ + \ & n \cdot 10 \cdot 0{,}4\ \mu s \text{ (istruzione di salto)} \\ + \ & 4 \cdot 0{,}4\ \mu s \text{ (istruzione di alt)} \end{aligned}$$

Se anziché l'istruzione di salto condizionato JP cc,nn fosse stata usata l'istruzione JP nn (salto incondizionato, con codice operativo C3 anziché C2) il programma non sarebbe mai uscito dal loop. La stessa cosa accadrebbe se nn = 011C (ad ogni salto il registro C viene ricaricato).

Lo stesso programma può essere infine scritto utilizzando al posto dell'istruzione JPNZ l'istruzione DJNZ,e.

Infatti quest'ultima istruzione è simile alle istruzioni di salto condizionato a parte il fatto che per determinare il salto si usa il valore contenuto nel registro B. Il contenuto di B infatti viene decrementato, e fintanto che esso è diverso da zero al PC (program counter) viene sommato il valore dello spostamento e (spiazzamento. L'assembler calcola lo spostamento relativo ed esegue un aggiustamento di e in modo da compensare il fatto che, essendo questa una istruzione a due byte, al termine della sua esecuzione il PC sarà stato automaticamente incrementato due volte. L'istruzione DNJZ,e è una istruzione a due byte:

00010000	= 10	(codice operativo)
← e-2 →	e-2	(Spiazzamento con codice complemento a due)

Nessun bit di condizione viene alterato. Il tempo d esecuzione è pari a $13 \cdot 0{,}4\ \mu s$ se B $\neq$ 0 e $8 \cdot 0{,}4\ \mu s$ se B = 0.

Nel caso specifico

$$e - 2 = -3 = FD$$

perché il numero di byte di spiazzamento per tornare all'indirizzo 0116 è uguale a -1 (0116-0117 = -1).

Il programma è il seguente:

Locazione di memoria	Codice oggetto	Assembly	Commenti
0114	06 n	LD B,n	carica n in B
0116	00	N0P	nessuna operazione
0117	10 FD	DJNZ,e	salta all'indirizzo specificato dal PC e dallo spiazzamento se il flag Z è zero (B $\neq$ 0)
0119	76	HALT	

Se il salto invece deve essere effettuato in avanti, ad esempio di tre byte (indirizzo di arrivo-indirizzo di partenza = 3 byte), l'istruzione corrispondente vale:

10 01 in quanto in questo caso e-2 = 3-2 = 1 = 01

L'istruzione N0P (nessuna operazione), il cui codice operativo è 00, fa si che durante questo ciclo macchina la CPU non esegua nessuna operazione. Questa istruzione viene spesso usata dal programmatore o per ritardare il programma oppure per annullare delle istruzioni senza dover scrivere di nuovo il programma in quanto le istruzioni di salto dovrebbero essere modificate nei loro operandi. Nessun bit di condizione viene alterato da questa istruzione. Il tempo di esecuzione è di 1,6 μs.

Loop di delay (ritardo)

Spesso accade di dover ritardare l'esecuzione di un programma di un certo ammontare di tempo. Ciò può essere fatto abbastanza semplicemente mediante uno o più loop di delay come mostrato nelle fig. 11 e 12.

Il massimo ritardo raggiungibile da un solo loop si ha quando il registro r è caricato con l'esadecimale FF (fig. 11).

Nel caso di fig. 12 il ritardo massimo raggiungibile si ottiene quando tutti e tre i registri utilizzati sono caricati con FF in quanto in questo caso il registro r_2 si decrementa di una unità ogni 256 decrementi del registro r_3. Analogamente il registro r_1 si decrementa di una unità ogni 256 decrementi del registro r_2.

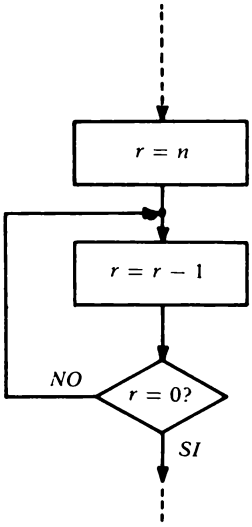


FIG. 11. - Semplice loop di delay.

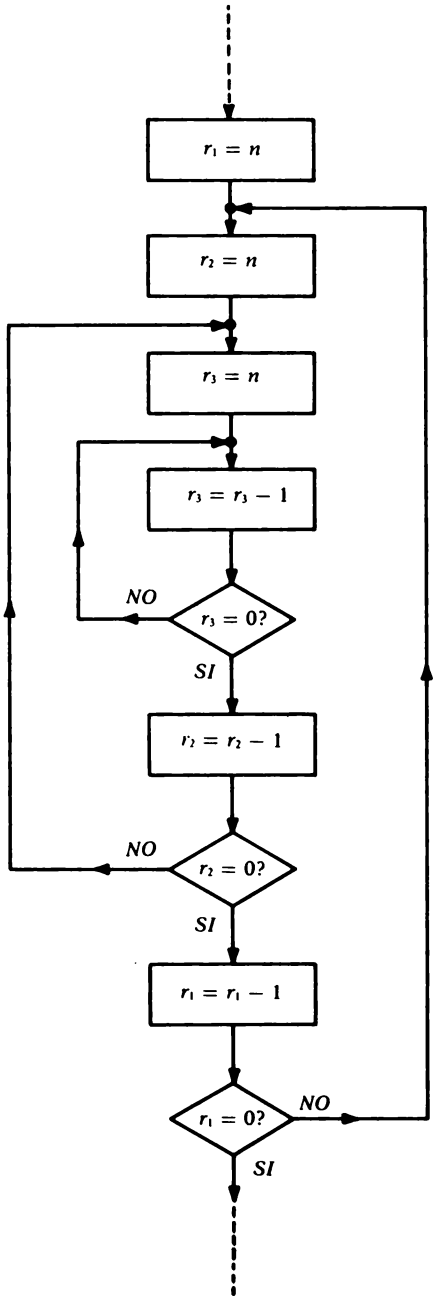


FIG. 12. - Tre loop di delay uno dentro l'altro.

Posizionamento ad 1 del bit di peso 2^2 di un byte

In questo esempio saranno introdotte tre nuove istruzioni.

Il problema da risolvere è il seguente:
dato il numero esadecimale C0 dividerlo per due fino a settare il bit B_2. Porre il risultato in un registro e contare il numero delle divisioni effettuate.

Flow chart:

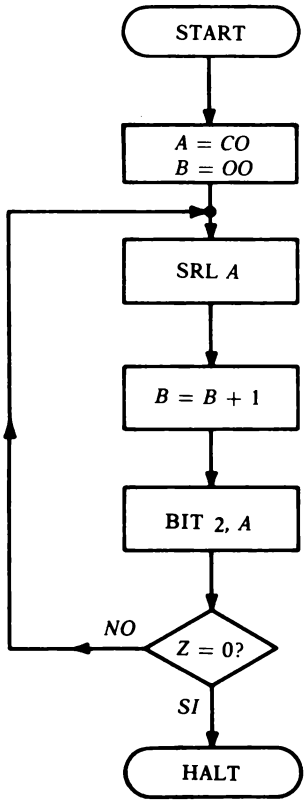


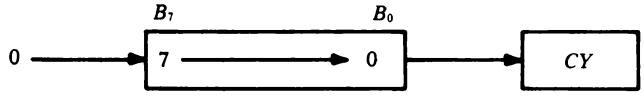
FIG. 13.

Nella stesura del programma si incontrano, oltre all'istruzione di caricamento di un registro, le istruzioni SRL s, INC r e BIT b,r.

La prima è una fra le tante istruzioni di scorrimento o shift, che nel caso riguardante registri da otto bit è costituita da due byte:

11001011 = CB
00111 r = 3F (nel caso del registro A, r = 111).

Questa istruzione sposta il contenuto del registro s verso destra; il contenuto del bit B_0 è copiato nel flag di carry ed il bit B_7 è messo a zero:



Cosicché se il registro A contiene 11000000 = C0, dopo l'istruzione SRL A il contenuto dell'accumulatore e del flag di carry sarà:

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	C
0	1	1	0	0	0	0	0	0

Cioè ogni scorrimento verso destra divide il byte per due. Analogamente uno scorrimento verso sinistra lo moltiplicherà per due.

Tutti i flags sono alterati dall'istruzione ed il tempo di esecuzione è pari a 3,2 μs.

L'istruzione INC r, ad un solo byte, incrementa di una unità il contenuto di un registro:

00 r 100

Il numero di cicli di clock per eseguirla è uguale a quattro. Tutti i flags ad eccezione di quello di carry sono alterati da questa istruzione.

L'istruzione BIT b,r mette nel flag di Z il complemento del bit testato del registro. Il formato dell'istruzione, a due byte, è il seguente:

11001011 = CB
01 b r = 57 (se il bit testato è B_2 ed il registro è l'accumulatore, b = 010 r = 111).

Sono necessari otto cicli di clock per eseguire l'istruzione. L'unico flag che rimane inalterato è quello di carry.

Il programma è il seguente:

Locazione di memoria	Codice oggetto	Assembly	Commenti
011A	3E C0	LD A,C0	carica il numero da dividere
011C	06 00	LD B,00	azzera il contatore di divisioni
P_1 011E	CB 3F	SRL A	dividi per due
0120	04	INC B	incrementa B
0121	CB 57	BIT 2,A	testa il bit 2 di A
0123	CA 1E 01	JPZ P_1	Salta a P_1 se Z = 1 (ciò avviene fin tanto che il bit 2 vale 0)
0126	76	HALT	

In sostituzione dell'istruzione JP Z è possibile utilizzare l'istruzione a due byte JR Z, e (salto relativo con spiazzamento complemento a due) il cui codice oggetto nel caso presente vale 28 F9.

Una volta eseguito il programma il registro A conterrà il risultato della divisione (0C), mentre il registro B il numero delle divisioni effettuate (04). Lo stesso risultato (fig. 14) poteva essere raggiunto, come può essere facilmente verificato, mediante l'istruzione di confronto CP s e l'introduzione di una opportuna parola di *maschera* nel registro C.

L'istruzione CP s, ad un solo byte, confronta il contenuto dell'operando s con il contenuto dell'accumulatore:

$$10111\ r = B9 \text{ se } s = C = 001$$

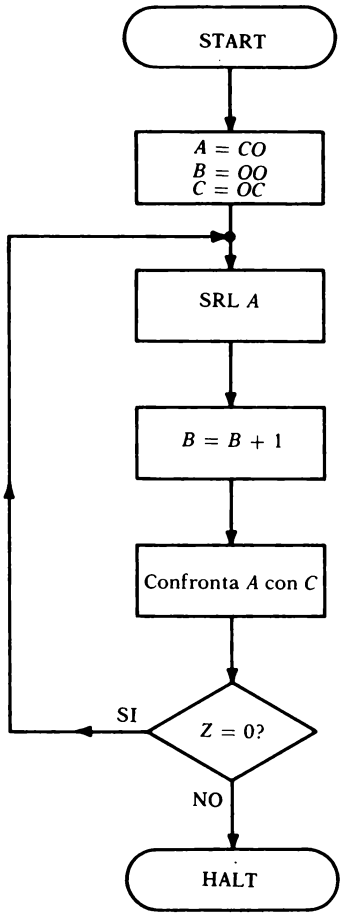


FIG. 14.

5. Sottoprogramma o subroutine

Quando una stessa sequenza di istruzioni compare diverse volte in un programma conviene separarla dal programma stesso e richiamarne l'esecuzione mediante una sola istruzione tutte le volte che il programma prin-

cipale lo richieda. Questa sequenza viene definita *sottoprogramma o subroutine* (fig. 15).

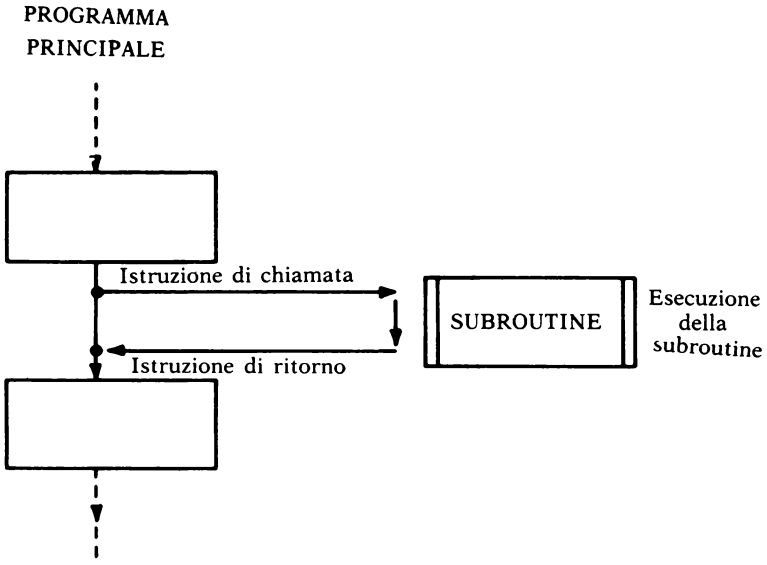


FIG. 15.

È chiaro che ogni volta che viene chiamata una subroutine deve essere memorizzato mediante un apposito registro specializzato (stack-pointer = registro puntatore) il punto del programma principale a cui si deve far ritorno una volta eseguito il sottoprogramma. Saranno qui di seguito esaminate quattro istruzioni molto importanti che vengono spesso usate nella stesura di programmi comprendenti subroutine. Esse sono:

CALL nn; RET; PUSH qq; POP qq

Istruzioni CALL e RET

Come si è già accennato nel precedente capitolo, nei sistemi a microprocessori più moderni come area di stack si utilizza una RAM esterna alla CPU di lunghezza arbitraria. Il registro specializzato che la indirizza è lo stack-pointer (SP).

L'area di stack può servire per immagazzinare temporaneamente dei dati, ma principalmente viene utilizzata per il salvataggio del program-counter nelle interruzioni e nelle chiamate di subroutine.

L'istruzione CALL nn è una istruzione a tre byte:

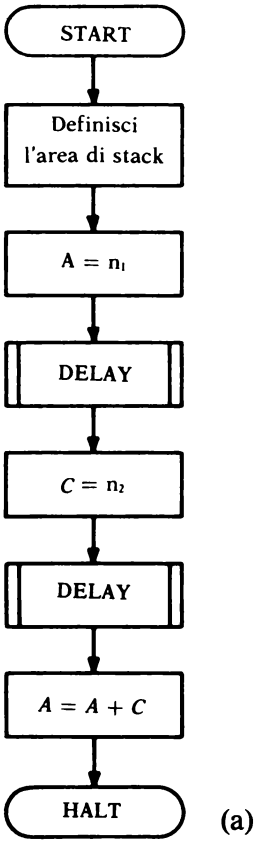
11001101	=	CD
← n →		indirizzo basso di memoria dello stack
← n →		indirizzo alto di memoria dello stack

ed esegue le seguenti operazioni:
dopo aver caricato il contenuto del PC nella parte alta dello stack, gli operandi nn vengono caricati nel PC per indicare l'indirizzo di memoria da cui deve essere estratto il codice operativo della subroutine. Alla fine del sottoprogramma viene usata l'istruzione RET (Return) per ritornare all'esecuzione del programma principale rimettendo la parte alta dello stack nel PC. Il caricamento avviene secondo la sequenza:

$$(SP\text{-}1) \leftarrow PC_H,\ (SP\text{-}2) \leftarrow PC_L,\ PC \leftarrow nn$$

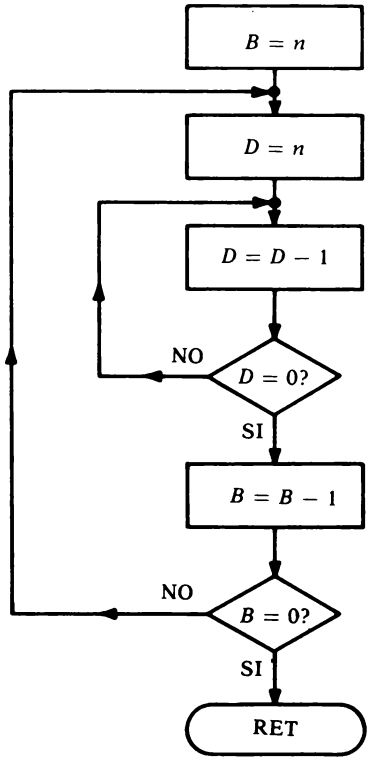
Lo stack-pointer viene decrementato e la parte alta di PC viene caricata nell'indirizzo da esso specificato. Lo SP viene decrementato di nuovo e la parte bassa del PC caricata nella parte alta dello stack. Il seguente esempio servirà da chiarimento:

Flow-chart main program:



(a)

Subroutine delay



(b)

FIG. 16. - Programma principale (a) e sottoprogramma (b).

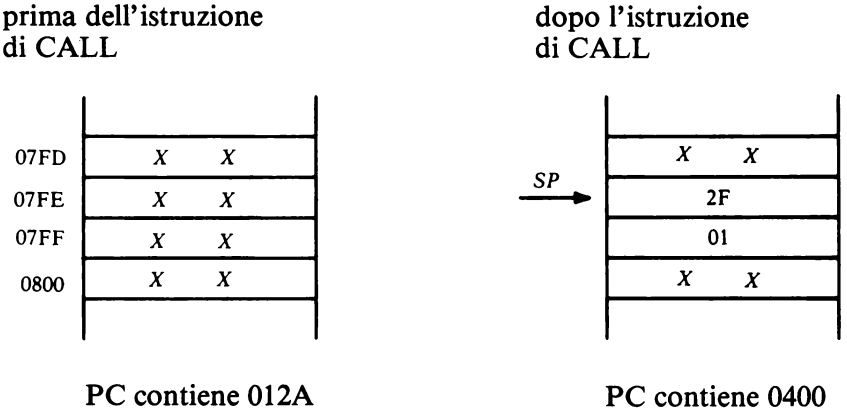
Programma:

Locazione di memoria	Codice oggetto	Assembly	Commenti
0127	31 00 08	LDSP,0800	definisci l'area di stack
012A	3E n_1	LD A,n_1	carica A con n_1
012C	CD 00 04	CALL DELAY	chiama la subroutine caricata a partire da 0400
012F	0E n_2	LD C, n_2	carica C con n_2
0131	CD 00 04	CALL DELAY	
0134	81	ADD A,C	somma A con C
0135	76	HALT	

Subroutine DELAY:

	0400	06 n	LD B,n
P_1	0402	16 n	LD D,n
P_2	0404	15	DEC D
	0405	C2 04 04	JPNZ P_1
	0408	05	DEC B
	0409	C2 02 04	JPNZ P_2
	040C	C9	RET

Come si modifica lo stack:



PC contiene 012A PC contiene 0400

FIG. 17.

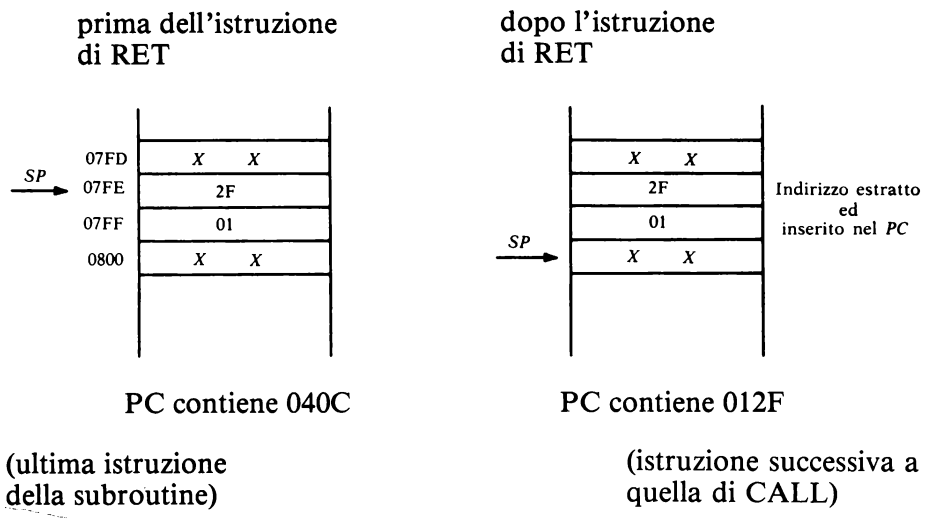


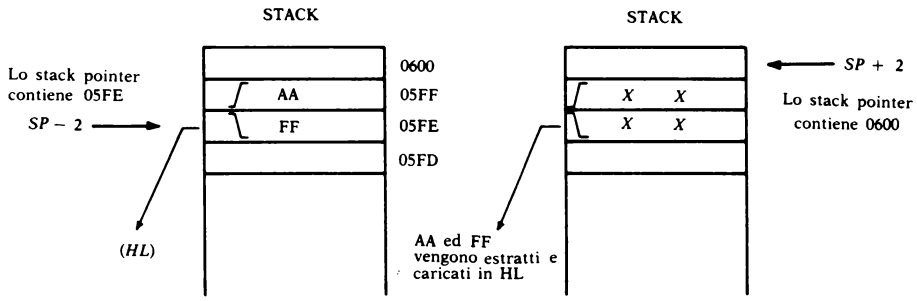
FIG. 18.

Nelle istruzioni di CALL nn e RET non viene alterato il contenuto del registro dei flags. I tempi di esecuzione sono uguali rispettivamente a 6,8 μs e 4μs.

Oltre alle istruzioni incondizionate di CALL nn e RET, il set di istruzioni dello Z 80 contiene quelle condizionate CALL cc, nn e RET cc. Quest'ultime chiamano la subroutine e permettono il ritorno al programma principale se la condizione cc è vera.

Istruzioni di PUSH qq e POP qq

Spesso accade nei programmi contenenti sottoprogrammi di dover far uso di un numero di registri superiore a quello disponibile dalla CPU. Per ovviare a questo inconveniente gli stessi registri usati nel programma principale (main program) possono essere utilizzati nell'esecuzione di subroutines avendo però l'accortezza di salvare il loro contenuto nell'area di stack precedentemente definita dall'operatore. Nello Z 80 le istruzioni che permettono il salvataggio ed il ripristino dei contenuti dei registri sono indicate con PUSH e POP rispettivamente. In fig. 19 è mostrata l'esecuzione delle istruzioni PUSH HL e POP HL, cioè il salvataggio e il reintegro del contenuto della coppia di registri HL, supponendo di aver caricato inizialmente lo stack-pointer con 0600 e HL con AAFF.



dopo l'esecuzione di PUSH dopo l'esecuzione di P0P

FIG. 19.

Programma:

Locazione di memoria	Codice oggetto	Assembly	Commenti
0136	31 00 06	LD SP,0600	definisci l'area di stack
0139	21 FF AA	LD HL,AAFF	carica in HL l'esadecimale AAFF
013C	E5	PUSH HL	salva il contenuto di HL nell'area di stack
013D	21 00 00	LD HL,0000	azzerra HL
0140	E1	P0P HL	ripristina il contenuto di HL
0141	76	HALT	

Al solito i codici oggetto esadecimali sono stati ricavati mediante l'uso della carta di riferimento tenendo presente che il caricamento di una coppia di registri con un intero a due byte fa capo all'istruzione LD dd,nn (dd definisce la coppia di registri ed nn l'intero da caricare).

Istruzioni LD r, (IX + d), NEG, AND s

L'istruzione LD r, (IX + d) carica il contenuto (espresso dalle parentesi) della locazione di memoria indirizzata da IX + d, con d numero intero, nel registro r. Il formato dell'istruzione è il seguente:

11011101 = DD
01 r 110
← d →

Il tempo di esecuzione è pari a 7,6 μs, il registro di flags risulta inalterato dopo questa istruzione. Se ad esempio il registro indice IX contiene l'esadecinale 0004, l'istruzione LD A, (IX + 06) esegue la somma di 0004 con 0006 puntando alla locazione di memoria 000A. Se quest'ultima locazione contiene il dato 0F, tale dato sarà caricato nell'accumulatore.

L'istruzione NEG complementa a due il contenuto dell'accumulatore. Se ad esempio prima di NEG l'accumulatore conteneva 01101100, dopo tale istruzione contiene 10010100. Il codice oggetto è ED 44, il tempo di esecuzione dell'istruzione uguale a 3,2 μs, i bit di condizione sono tutti alterati.

L'istruzione AND s infine, esegue l'AND logico tra l'operatore s e l'accumulatore. Nel caso che s sia un registro il codice operativo vale:

10100 r

Tutti i flags sono alterati dall'istruzione, il tempo di esecuzione è pari a 1,6 μs.

Istruzioni OUT (C), r ed RST p

L'istruzione OUT (C), r pone nella parte bassa degli indirizzi (da A_0 ad A_7) il contenuto del registro per poter così selezionare un dispositivo di ingresso-uscita (da 00 a FF) mentre il contenuto del registro B viene messo nella parte alta degli indirizzi.

Il contenuto del registro r è messo nel bus dei dati e scritto nel dispositivo di I/0 selezionato. L'istruzione non altera il registro di flags, ha un tempo di esecuzione di 4,8 μs ed è formata da due byte:

11101101 = ED
01 r 001

Le istruzioni di Restart (RST p) sono considerate come particolari istruzioni di chiamata a subroutine. Il formato di queste istruzioni è ad un solo byte:

11 ← t → 111

dove t indica una configurazione binaria compresa tra 000 e 111.

Il contenuto del PC viene salvato nello stack (come nelle istruzioni di CALL) permettendo così un rientro al programma principale al termine della subroutine di servizio che dovrà concludersi con un RET:

Il controllo del programma è trasferito all'istruzione contenuta all'indirizzo calcolato nel modo seguente:

00000000	00 t 000
(HI)	(L0)

Se ad esempio il contenuto del PC è 0200 dopo l'esecuzione della istruzione RST 38 il PC conterrà 0038 come l'indirizzo del codice operativo da prelevare successivamente. Dalla carta di riferimento, nel caso di p = 38 (t = 111), all'istruzione RST 38H corrisponde il codice operativo FF. Il

tempo di esecuzione è pari a 4,4 μs, nessun bit di condizione viene alterato.

Solitamente le istruzioni di Restart sono usate in stretta relazione con le otto routine (lunghe ognuna otto byte) allocate nelle prime 64 parole di memoria per servire richieste di interrupt (interruzione) provenienti dall'esterno (cap. 4°). Se la routine di servizio, il cui indirizzo di partenza è stato richiamato attraverso la istruzione di Restart, richiede più di otto byte per funzionare, al suo indirizzo verrà previsto un altro salto ad una subroutine collocata in una zona diversa di memoria.

Sono qui di seguito riportati il flow-chart ed il programma inerenti alla selezione ripetuta di un dispositivo di uscita il cui codice è stato caricato nel registro C:

Flow-chart:

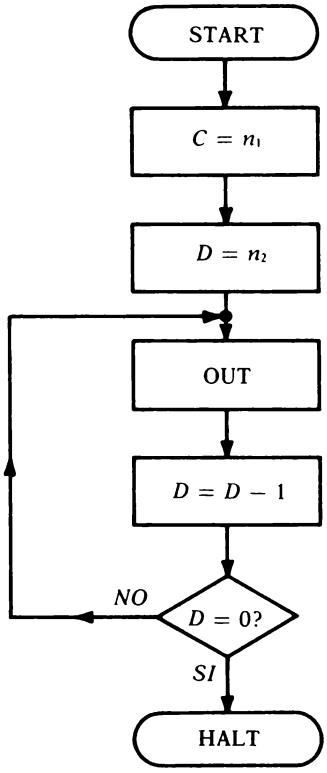


FIG. 20.

Programma:

Locazione di memoria	Codice oggetto	Assembly	Commenti
0142	0E n_1	LD C, n_1	carica il codice dispositivo di uscita (00 $\leqq$ n_1 $\leqq$ FF)
0144	16 n_2	LD D, n_2	carica con n_2 il contatore di selezione
P_1 0146	ED 59	OUT(C),E	seleziona il dispositivo n_1
0148	15	DEC D	decrementa il contenuto del registro D
0149	C2 46 01	JPNZ P_1	Salta a P_1 se D $\neq$ 0
014C	FF	RST 38 H	controllo alla CPU

6. Metodi di indirizzamento dello Z 80

A volte accade che un microprocessore avente un set di istruzioni molto contenuto riesca ad eseguire un certo programma con un minor numero di istruzioni e occupare una più piccola zona di memoria rispetto ad un altro microprocessore con un set di istruzioni più vasto e quindi a prima vista più potente. Ciò è dovuto ad una maggiore flessibilità di indirizzare la memoria o i registri di lavoro, e quindi ai metodi di *indirizzamento usati.*

I metodi di indirizzamento dello Z 80, già utilizzati negli esempi delle pagine precedenti, sono:

- Indirizzamento immediato -

L'operando è contenuto direttamente nel secondo byte dell'istruzione. Esempio:

LD r,n 00 r 110
← n →

:carica il dato n nel registro r.

- Indirizzamento diretto -

Il secondo e terzo byte dell'istruzione contengono rispettivamente la parte bassa ed alta dell'indirizzo.

Esempio:

LD (nn), A 00110010 = 03
← n →
← n →

: carica l'accumulatore nella locazione puntata dall'indirizzo nn.

- Indirizzamento indiretto tramite coppie di registri -

Nella coppia di registri utilizzata è contenuto l'indirizzo del dato richiesto.
Esempio:

LD A, (DE) 00011010 = 1A

: carica l'accumulatore con il contenuto della locazione puntate da DE.

- Indirizzamento indicizzato -

È simile a quello con coppie di registri ad eccezione dell'esistenza dello spiazzamento d.

Esempio:

LD r, (IY + d) 11111101 = FD
01 r 110
← d →

: carica il registro r con il contenuto della locazione puntata da IY + d.

- Indirizzamento tramite stack-pointer

L'indirizzo è contenuto nello stack pointer.
Esempio:

PUSH qq 11 qq 0101

(con qq coppia di registri fra BC, DE, HL o AF).

: carica la coppia di registri qq nella parte alta dello stack.

- Indirizzamento mediante istruzioni di salto relativo -

Viene fornito l'indirizzo della locazione di memoria da cui riparte il programma.
Esempio:

JR e 00011000 = 18
← e-2 →

salta all'indirizzo puntato da PC + e in modo incondizionato.

Utilizzando opportunamente questi metodi di indirizzamento l'operatore potrà raggiungere una ottimizzazione dei programmi sia per quanto riguarda il numero delle istruzioni sia per la parte di memoria occupata.

7. Esercitazioni di software

In questo paragrafo saranno svolti alcuni esempi di programmi (non sempre ottimizzati) leggermente più complicati, rispetto a quelli appena visti, facenti uso in parte di nuove istruzioni. I flow-chart, così come il commento dei singoli passi, non sono spiegati dettagliatamente nelle loro funzioni così come è stato fatto precedentemente. Ciò con l'intento di far verificare al lettore, attraverso una attenta analisi dei problemi proposti, l'esattezza degli algoritmi scelti e familiarizzarlo maggiormente con la carta di riferimento portandolo così ad effettuare una ottimizzazione dei programmi dove è possibile.

Addizione di tre numeri

Il programma esegue la somma di tre numeri di otto bit. Il primo ed il secondo numero sono contenuti rispettivamente nell'accumulatore e nel registro C, il terzo nella locazione di memoria 0200. Il risultato dell'addizione è salvato nello stack e così anche il registro di flag posizionato in base al risultato della somma.

Flow-Chart:

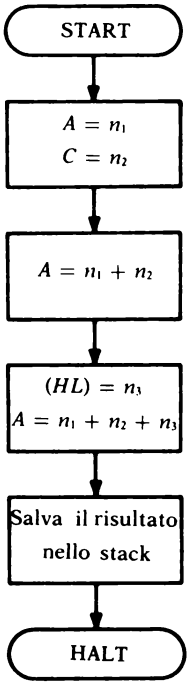


FIG. 21.

Programma:

Locazione di memoria	Codice oggetto	Assembly	Commenti
0100	3E n_1	LD A, n_1	
0102	0E n_2	LD C, n_2	
0104	81	ADD A,C	somma il primo ed il secondo numero
0105	21 00 02	LD HL, 0200	
0108	36 n_3	LD (HL), n_3	
010A	86	ADD A, (HL)	somma i tre addendi
010B	31 00 08	LD SP, 0800	definisci l'area di stack
010E	F5	PUSH AF	salva il contenuto dei registri A e F
010F	76	HALT	

Se ad esempio n_1 = 0D, n_2 = 08, n_3 = FA al termine del programma la locazione di memoria 07FF conterrà 0F (accumulatore) quella di indirizzo 07FE il dato 09 (registro di flags, parag. 3):

0D	00001101	
+ 08	+ 00001000	
15	00010101	
+ FA	+ 11111010	
= 0F	= 1 00001111	contenuto dell'accum.
09	0 0 0 0 1 0 0 1	Contenuto del registro
	S Z x H x P/V N C	di flag.

Livelli di subroutine

Una volta caricati quattro dati a partire dall'indirizzo di memoria 0300 si vuole calcolare la differenza tra il primo e il secondo, tra tale differenza parziale ed il terzo dato, tra questo ultimo e il quarto dato. La differenza finale deve essere salvata alla locazione di memoria 0304.

Il programma che esegue il problema proposto è strutturato in un programma principale (main program) che provvede alle inizializzazioni e alla acquisizione dei quattro dati, e da subroutines di 1° e 2° livello (fig. 22) che eseguono la differenza tra i due dati.

La subroutine di 1° livello (SUB) chiamata dal programma principale esegue la differenza tra due dati e nel caso di risultato negativo lo complementa a due. La complementazione a due è eseguita dalla subroutine di 2° livello (COMP) richiamata da quella del 1° se necessario.

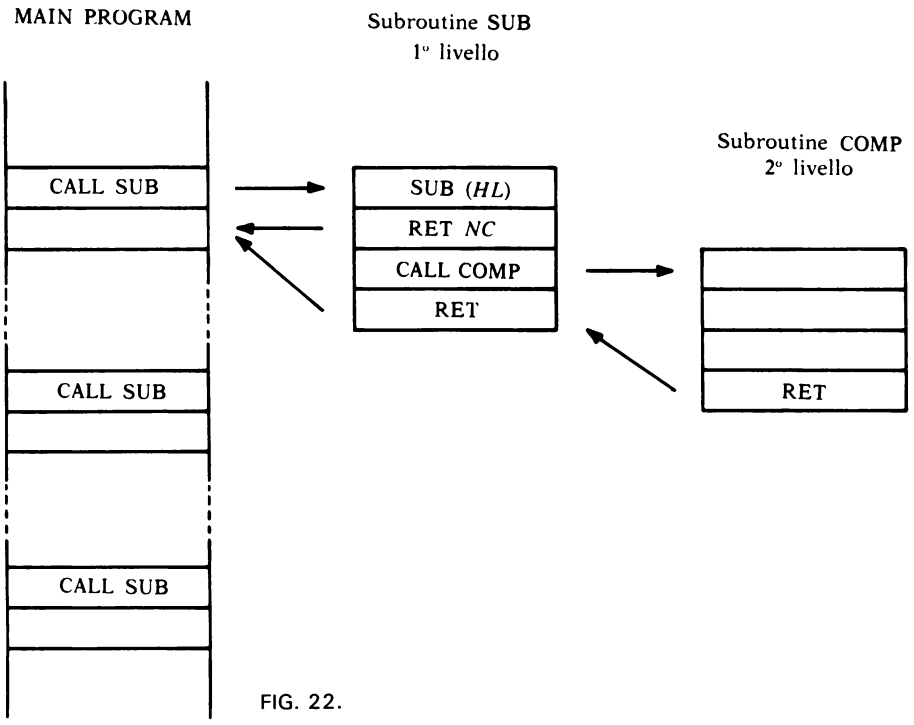


FIG. 22.

Programma principale:

Locazione di memoria	Codice oggetto	Assembly	Commenti
0200	31 00 08	LD SP, 0800	inizializza lo stack-pointer
0203	21 00 03	LD HL, 0300	inizializza HL
0206	7E	LD A, (HL)	acquisisci il 1° e il 2° dato
0207	23	INC HL	
0208	CD 00 05	CALL SUB	sottrai
020B	23	INC HL	
020C	CD 00 05	CALL SUB	sottrai alla differenza precedente il 3° dato
020F	23	INC HL	
0210	CD 00 05	CALL SUB	sottrai alla differenza precedente il 4° dato
0213	23	INC HL	
0214	77	LD (HL), A	salva il risultato
0215	76	HALT	

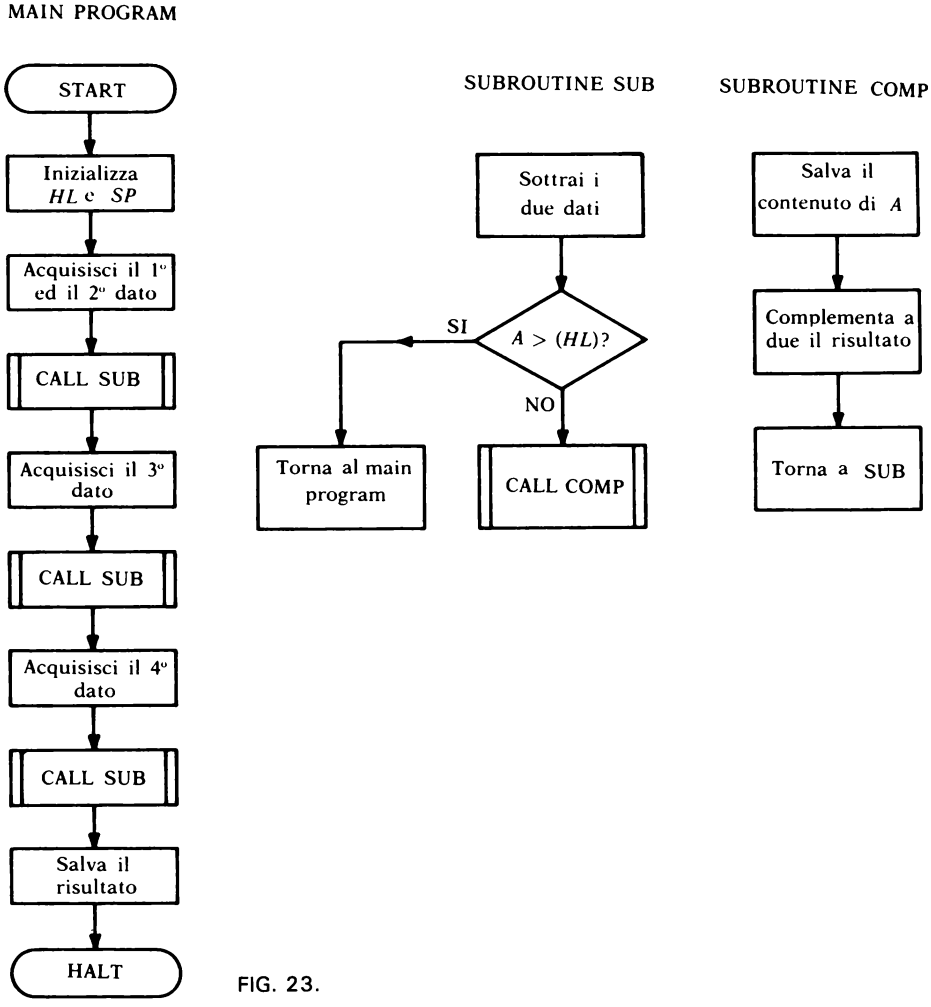


FIG. 23.

Subroutine SUB:

0500	96	SUB (HL)	sottrai il contenuto della locazione di memoria puntata da HL dall'accumulatore
0501	D0	RET NC	se (A) > (HL) ritorna al programma principale
0502	CD 00 06	CALL COMP	se (A) < (HL) chiama COMP
0505	C9	RET	

Subroutine COMP:

0600	32 05 03	LD (0305), A	salva il contenuto di A
0603	2F	CPL	complementa a due l'accumulatore
0604	C6 01	ADD A, 01	
0606	C9	RET	ritorna a SUB

Riempimento di una area di memoria con un dato costante (Fill)

Il seguente programma provvede a riempire con un dato costante una area di memoria di ampiezza non superiore a 255 byte.

Il numero delle locazioni di memoria da scrivere è caricato nell'accumulatore, mentre l'indirizzo di partenza è impostato nella coppia di registri HL. In questo caso il dato costante BB viene caricato inizialmente nella locazione di memoria 0400 per poi essere scritto a partire dall'indirizzo

Flow-chart:

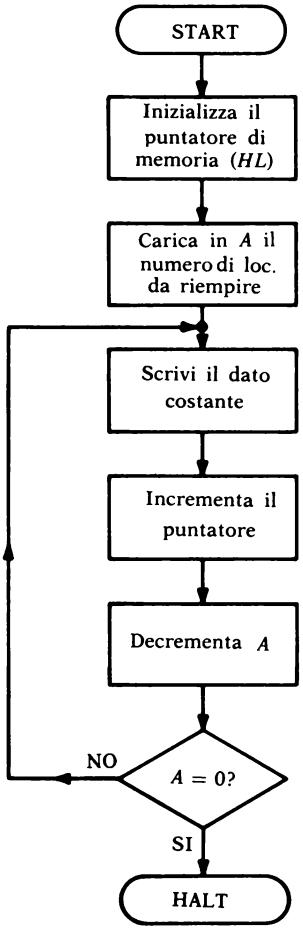


FIG. 24.

0300. Il contenuto dell'accumulatore è stato posto uguale ad FF.A fine programma il dato BB sarà presente nelle locazioni di memoria che vanno dall'indirizzo 0300 fino a 03FE compreso. Resta inteso che è sempre possibile variare da programma sia il dato costante da scrivere, sia modificare l'indirizzo e l'ampiezza dell'area di memoria da riempire.

Programma:

Locazione di memoria	Codice oggetto	Assembly	Commenti
01FB	21 00 03	LD HL, 0300	punta alla locazione 0300
01FE	3E FF	LD A, FF	carica A con il numero delle locazioni da riempire
P_1 0200	36 BB	LD (HL), BB	scrivi il dato nella locazione puntata da HL
0202	23	INC HL	
0203	3D	DEC A	
0204	C2 00 02	JPNZ P_1	È caricato il dato in tutte le 255 locazioni?
0207	76	HALT	

Ordinamento in forma crescente di 256 numeri

Il presente programma esegue l'ordinamento in forma crescente di 256 numeri residenti in memoria a partire da un certo indirizzo (in questo esempio 0500). Il procedimento è il seguente:

— si esaminano il primo e il secondo numero e, se necessario, vengono ordinati in forma crescente;

— si procede similmente per il secondo e terzo numero, il terzo ed il quarto e così via.

Questa prima fase non è però sufficiente per l'ordinamento desiderato, infatti, facendo riferimento all'ordinamento dei primi tre numeri, nel caso peggiore si avrà:

02	01	01	
01	02	00	
00	00	02	risultato del primo ciclo di ordinamento.

A questo punto si ripete l'ordinamento partendo dal risultato precedente:

01	00	
00	01	
02	02	risultato del secondo ed ultimo ciclo di ordinamento

Di conseguenza per ordinare in forma crescente tre numeri disposti inizialmente nella situazione più sfavorevole si sono resi necessari due cicli di ordinamento a due a due.

Per ordinare quindi 256 numeri in forma crescente occorrerà predisporre 255 cicli (n-1 = 255, con n numeri da ordinare) uguali a quello appena visto (contatore di cicli). Inoltre all'interno di ogni ciclo debbono essere ordinate 255 coppie di numeri (contatore di coppie).

Programma:

	Locazione di memoria	Codice oggetto	Assembly	Commenti
	0300	16 FF	LD D, FF	contatore di cicli = 255
P_1	0302	21 00 05	LD HL, 0500	punta al 1° numero da ordinare
	0305	0E FF	LD C, FF	contatore di coppie = 255
P_2	0307	7E	LD A, (HL)	
	0308	23	INC HL	
	0309	46	LD B, (HL)	acquisisci e confronta la coppia
	030A	B8	CP B	
	030B	DA 12 03	JP C, P_3	se la coppia è ordinata salta a P_3 altrimenti ordinala
	030E	77	LD (HL), A	
	030F	2B	DEC HL	
	0310	70	LD (HL), B	Ordinamento
	0311	23	INC HL	
P_3	0312	0D	DEC C	
	0313	3E 00	LD A,00	
	0315	B9	CP C	tutte le coppie sono ordinate?
	0316	CA 1C 03	JP Z, P_4	
	0319	C3 07 03	JP, P_2	
P_4	031C	15	DEC D	
	031D	C2 02 03	JPNZ, P_1	tutti i cicli di ordinamento sono stati eseguiti?
	0320	76	HALT	

Flow-chart:

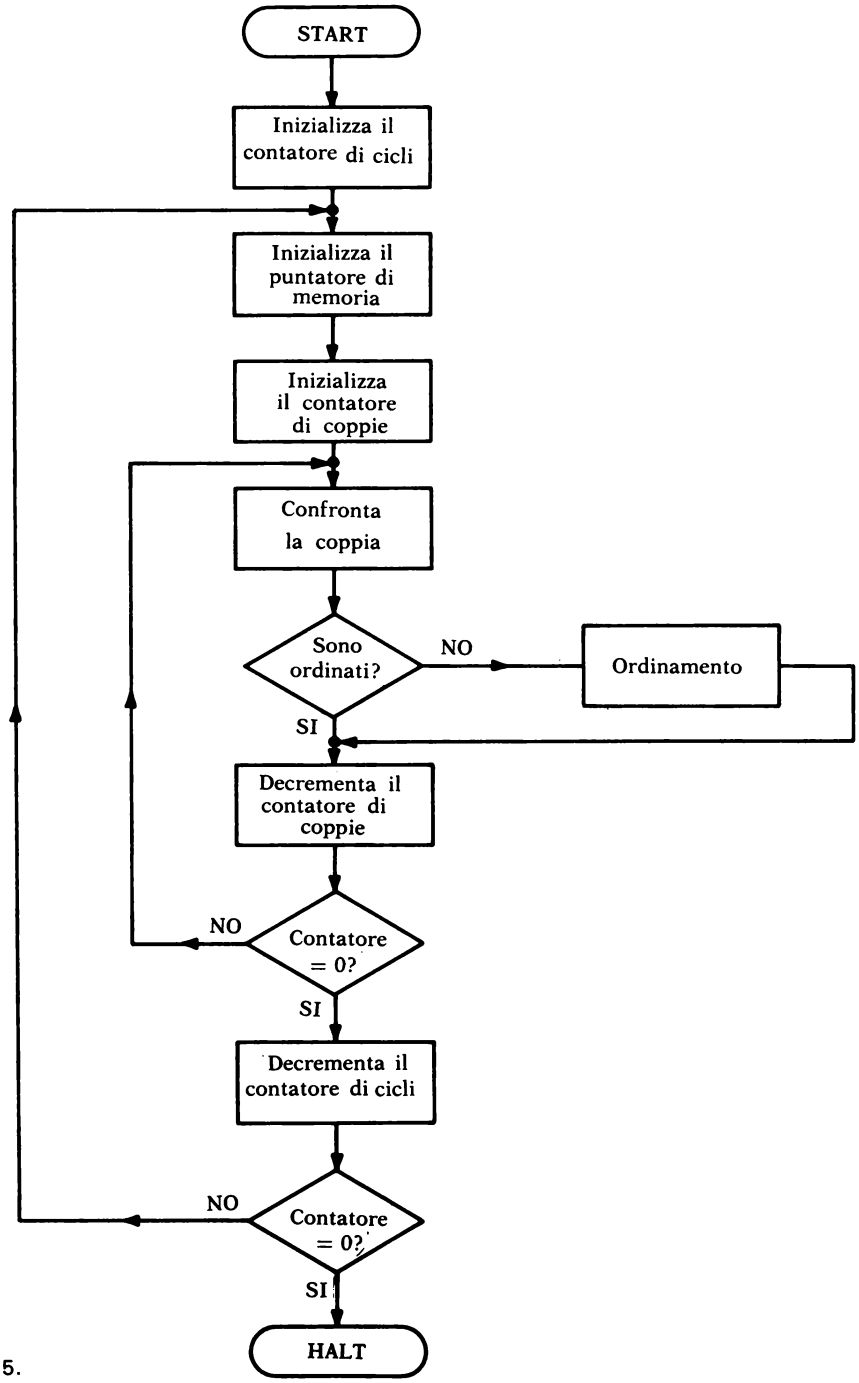


FIG. 25.

Conversione binario-BCD

Il seguente programma converte un numero binario espresso con 16 bit, contenuto nella coppia di registri HL, nel suo corrispondente BCD. Il risultato è posto in memoria a partire dall'indirizzo contenuto dalla coppia di registri DE (ad esempio 0242). La parte del programma che esegue la conversione è strutturata in subroutine ed è richiamata dal programma principale che provvede alle inizializzazioni ed a salvare in memoria il risultato finale. L'algoritmo della conversione consiste nel sottrarre al numero binario ($\leqq 65535_{10}$) la quantità $(10000)_{10}$ e contare il numero di sottrazioni necessarie che non portano ad un risultato negativo. Tale numero è la cifra più significativa in BCD (MSD). Per ottenere le cifre rimanenti è sufficiente procedere con analoghe sottrazioni per 1000_{10}, 100_{10}, 10_{10}.

Il resto dell'ultimo ciclo di sottrazioni rappresenta la cifra meno significativa (LSD). Infatti se per esempio il numero da convertire è A120 = 57376_{10} si ha:

57376	
— 10000	1ª sottrazione
47376	
— 10000	2ª sottrazione
37376	
— 10000	3ª sottrazione
27376	
— 100000	4ª sottrazione
17376	
— 10000	5ª sottrazione MSD
7376	
7376	
— 1000	1ª sottrazione
6376	
— 1000	2ª sottrazione
5376	
— 1000	3ª sottrazione
4376	
— 1000	4ª sottrazione
3376	
— 1000	5ª sottrazione

2376	
— 1000	6ª sottrazione
1376	
— 1000	7ª sottrazione
376	
376	
— 100	1ª sottrazione
276	
— 100	2ª sottrazione
176	
— 100	3ª sottrazione
76	
76	
— 10	1ª sottrazione
66	
— 10	2ª sottrazione
56	
— 10	3ª sottrazione
46	
— 10	4ª sottrazione
36	
— 10	5ª sottrazione
26	
— 10	6ª sottrazione
16	
— 10	7ª sottrazione
6	LSB

Cosicché A120 = 57376

Per la conversione quindi del numero 1010000100100000 basterà caricare nel programma, al posto di nn, l'esadecimale 20A1 nelle locazioni di memoria 0204 e 0205. Alla fine del programma la locazione di memoria 021D conterrà la cifra più significativa mentre la locazione di memoria 0221 quella meno significativa.

Programma:

Locazione di memoria	Codice oggetto	Assembly	Commenti
0200	31 00 08	LD SP,0800	definisci l'area di stack
0203	21 nn	LD HL, nn	carica il numero da convertire
0206	11 24 02	LD DE,0224	punta l'indirizzo del risultato
0209	01 F0 D8	LD BC,D8F0	BC = -10000_{10}
020C	CD 00 03	CALL CONV	
020E	01 18 FC	LD BC,FC18	BC = -1000_{10}
0212	CD 00 03	CALL CONV	
0215	01 9C FF	LD BC, FF9C	BC = -100_{10}
0218	CD 00 03	CALL CONV	
021B	01 F6 FF	LD BC, FFF6	BC = -10_{10}
021E	CD 00 03	CALL CONV	
0221	7D	LD A,L	
0222	12	LD (DE),A	memorizza la cifra meno significativa
0223	76	HALT	
CONV:			
0300	AF	XOR A	azzerra l'accumulatore
0301	D5	PUSH DE	salva il contenuto di DE
P_1 0302	5D	LD E,L	salva L in E
0303	54	LD D, H	salva H in D
0304	3C	INC A	
0305	09	ADD HL, BC	inizia le sottrazioni
0306	DA 02 03	JP C, P_1	salta a P_1 se il risultato è negativo
0309	3D	DEC A	
030A	6B	LD L,E	
030B	62	LD H,D	carica in HL il resto del ciclo di sottrazioni
030C	D1	P0P DE	ripristina il contenuto di DE
030D	12	LD (DE),A	carica il contenuto di A nella locazione di memoria puntata da DE
030E	13	INC DE	
030F	C9	RET	ritorna al programma principale

ESERCIZI PROPOSTI

1. È corretta la seguente sequenza di programma?

```
.
.

PUSH HL
PUSH DE
.
.
.
POP HL
POP DE
.
.
.
```

2. Perchè il seguente programma non è corretto?

```
015C  LD SP,0162
      LD HL,nn
      LD DE,nn
      LD BC,nn
      PUSH HL
      PUSH DE
      PUSH BC
            .
            .
            .
      POP BC
      POP DE
      POP HL
            .
            .
            .
0180  RST 38H
```

3. Memorizzare direttamente il dato AABB, contenuto nella coppia di registri HL, nelle locazioni di memoria 0100 e 0101.

4. Realizzare un loop di ritardo della durata di 0,5 s utilizzando i registri A,B.C.

5. Eseguire il prodotto logico tra i contenuti dei registri A e B salvando il risultato in C.

6. Porre sequenzialmente sul codice dispositivo 0020 dieci byte precedentemente memorizzati a partire dalla locazione di memoria 019A

7. Trasferire dieci byte caricati a partire dalla locazione di memoria 01AD nella zona di memoria con indirizzo iniziale 0400 utilizzando l'istruzione LDIR.

8. Ordinare in modo crescente sedici numeri esadecimali precedentemente memorizzati a partire dall'indirizzo 01C3.

9. Eseguire la moltiplicazione tra due numeri binari a 16 bit memorizzati nelle locazioni di memoria 01E7-01E8 e 01E9-01EA salvando il risultato nelle locazioni 01EB-01EC.

10. Codificare a partire dalla locazione di memoria 0220 il programma il cui flow-chart è quello di fig. 14.

CAPITOLO TERZO

L'HARDWARE DEL MICROPROCESSORE

1. Lo scambio di informazioni

Il problema fondamentale, per l'esecuzione di un programma in un elaboratore, è il trasferimento di informazioni sia tra i vari elementi costituenti l'interno della CPU che tra tutti gli altri componenti esterni quali ad esempio: memorie, tastiere, terminali video, eccetera.

È bene ricordare che nella presente trattazione, con il termine di informazione si intendono i dati, gli indirizzi ed i vari segnali di controllo che viaggiano sotto forma di impulsi elettrici lungo i fili di connessione o qualsiasi altro mezzo adatto al loro trasporto.

Una fondamentale distinzione in tali trasferimenti può farsi in base alle distanze che si devono coprire; si hanno così i collegamenti per lunghe distanze, nei quali i canali di comunicazione vengono arricchiti di apparecchiature anche di notevole complessità, che verranno trattati nel capitolo 5 e quelli per brevi distanze che saranno l'argomento del presente paragrafo.

Questi ultimi sono concepiti per la connessione di elementi di un unico sistema, quale ad esempio un microcomputer, e fisicamente sono realizzati con mezzi passivi come connettori e cavi.

Una ulteriore suddivisione può effettuarsi rispetto alle modalità di collegamento che, nel caso di sistemi a microprocessori, sono le seguenti:

a) trasferimento punto a punto
b) trasferimento a diffusione
c) trasferimento a collettore
d) trasferimento a bus.

a) trasferimento punto a punto

Nel trasferimento punto a punto, si ha una tramissione di informazioni da un solo elemento detto *Sorgente* (S) ad un solo elemento chiamato *Destinazione* (D), secondo lo schema di figura 1.

Le informazioni viaggiano in un solo verso e in tal caso l'insieme dei fili di collegamento è chiamato bus monodirezionale anche se questa definizione non è del tutto corretta poiché il nome di bus spetta solamente ai fili

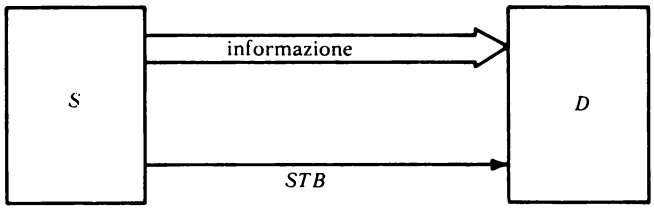


FIG. 1. - Trasferimento punto a punto.

di collegamento comuni a diverse sorgenti e diverse destinazioni contemporaneamente.

Poiché i dati vengono continuamente inviati dall'elemento S a quello D, il più delle volte senza rispettare una cadenza precisa, quest'ultimo deve conoscere con esattezza in quali istanti è presente in modo stabile un nuovo dato al suo ingresso. A questo compito provvede l'elemento sorgente mediante una linea di controllo della trasmissione che generalmente prende il nome di STROBE (STB).

Nella figura 2 è mostrato un esempio di successione temporale di trasferimento punto a punto.

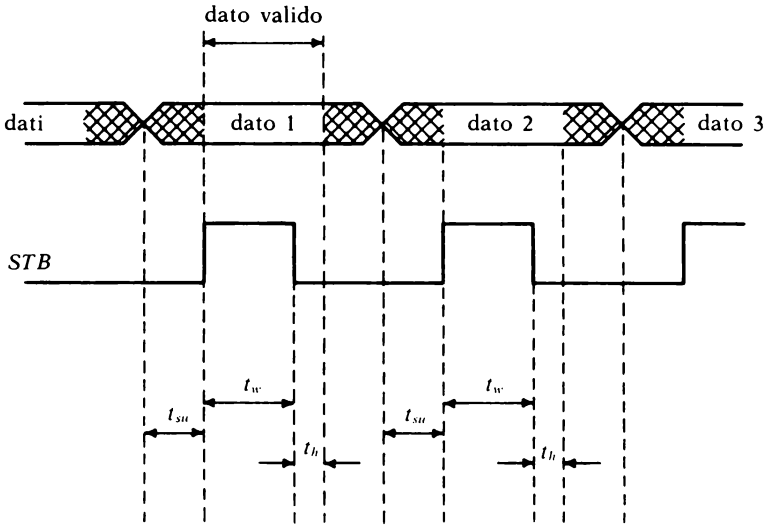


FIG. 2. - Flusso dei segnali nel trasferimento punto a punto.

Il segnale di STB, che abilita alla ricezione del dato l'elemento destinazione, viene inviato dall'elemento sorgente con un certo ritardo rispetto al dato, in modo tale che tutti i segnali elettrici che lo costituiscono abbiano il tempo di assestarsi al loro valore stabile. Questo tempo viene chiamato t_{su} (set up time = tempo di assetto) ed indica anche l'eventuale tempo ne-

cessario ai circuiti di decodifica di indirizzo per svolgere il loro compito.

Si tenga presente che le definizioni dei tempi che verranno via via date, pur cercando di essere le più generali possibile, possono discostarsi da quelle fornite dalle varie case costruttrici a causa della difformità dei loro manuali.

Lo strobe, deve invece annullarsi con un certo anticipo rispetto all'istante in cui i dati iniziano a cambiare, per tenere conto del tempo che impiega l'elemento D a disabilitarsi. Questo tempo viene chiamato t_h (hold time = tempo di ritenuta) e se fosse troppo lungo porterebbe a sovrapporsi il secondo dato al primo.

Con t_w (Write = scrittura) si indica il tempo minimo che deve avere l'impulso di STB per effettuare la scrittura.

ESEMPIO 1.

Si vuole trasferire una parola di 4 bit formata dalle uscite di un contatoire DM 7493, ad un elemento destinazione costituito da un latch quadruplo DM 7475, secondo lo schema di figura 3.

Il clock del sistema opera ad 1 MHz ed i tempi di propagazione fra ingressi ed uscite del contatore sono i seguenti:

DA INGRESSO	AD USCITA	t (ns)
A (clock)	QA	11
QA	QB	12
QB	QC	10
QC	QD	12

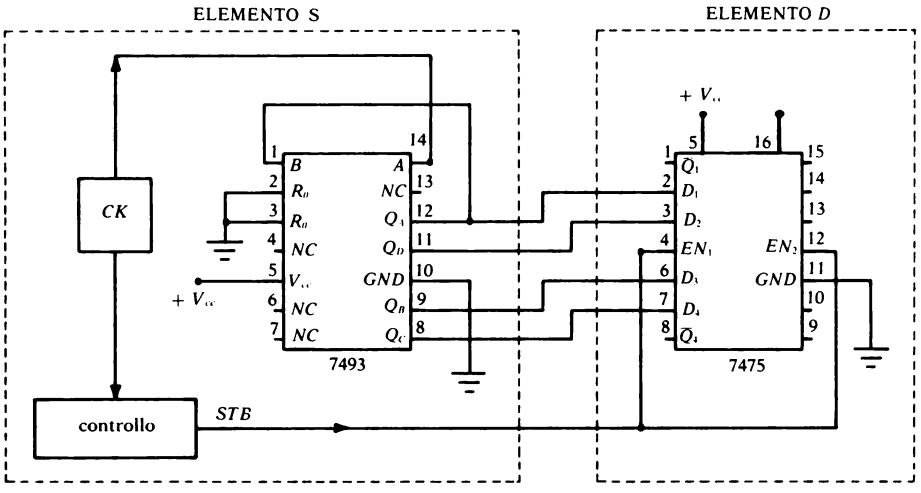


FIG. 3. - Esempio di trasferimento punto a punto.

Il diagramma temporale delle uscite del contatore risulta allora come quello di figura 4.

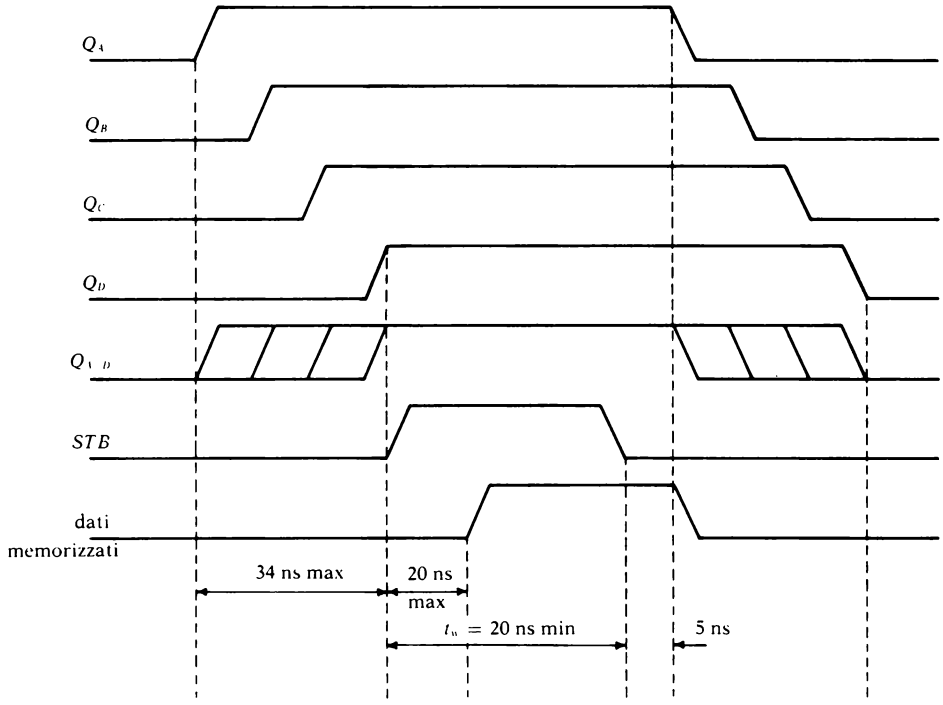


FIG. 4. - Diagramma temporale delle uscite del contatore.

Come si vede, i dati risultano stabili dopo un tempo di 34 ns (12 + 10 + 12), rispetto all'istante in cui compare il primo dato (Q_A).

L'impulso di STB, che l'elemento S deve inviare a quello D per abilitarlo alla memorizzazione, deve avere un ritardo di 34 ns, ma la memorizzazione inizia con un ulteriore ritardo di 16 ns che è il tempo di abilitazione del 7475.

Sempre dal manuale, si trae che il segnale di strobe deve avere una durata di almeno 20 ns (t_w = 20 ns) e disattivarsi 5 ns prima dell'inizio del cambiamento dei dati (t_h = 5 ns).

b) trasferimento a diffusione

Nel trasferimento a diffusione un solo elemento sorgente invia a più elementi destinazione, in tempi successivi, il dato.

Nella figura 5 è mostrato un esempio di una struttura di questo tipo. Le varie linee di STB, che abilitano di volta in volta il singolo elemento D alla

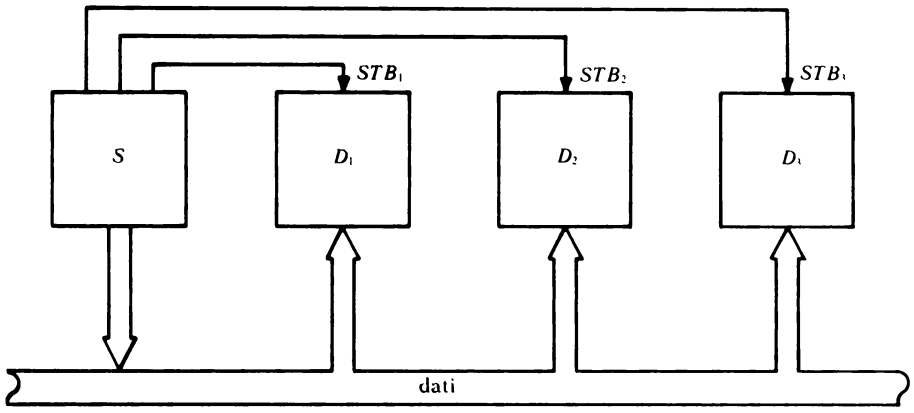


FIG. 5. - Trasferimento a diffusione.

ricezione dell'informazione, possono provenire o dall'elemento S stesso, come è mostrato in figura, oppure da un controllore separato. Come per il caso precedente il bus è monodirezionale.

ESEMPIO 2.

Si supponga di voler trasferire il contenuto di una memoria RAM 2101-A della Intel, in due memorie RAM 2112-A della Motorola, in modo alternativo: i dati con indirizzo pari in una e quelli con indirizzo dispari nell'altra, lasciando inalterato il loro indirizzo di partenza.

Entrambe le memorie sono statiche, in tecnologia MOS ed organizzate in 4 bit per 256 locazioni. Le loro principali caratteristiche a.c., delle quali ci si serve, sono le seguenti (figure 6 e 7):

Una prima difficoltà, che si incontrerà quasi sempre utilizzando manuali diversi in fase di progetto, è la grande differenza dei simboli e delle nomenclature utilizzate.

Nella figura 8 è mostrato uno dei circuiti di possibile realizzazione.

I due contatori 7493 posti in cascata, costituiscono il bus degli indirizzi che varia da 00000000 ad 11111111 su comando del clock principale.

La costituzione dell'elemento di controllo delle temporizzazioni, risulta abbastanza semplice, poiché le abilitazioni alle due memorie destinazione possono venire date direttamente dal bit meno significativo degli indirizzi, che quando vale 1 indica che il numero è dispari e viceversa.

Per dare tempo all'indirizzo di stabilizzarsi, vengono interposti dei buffer, uno invertente ed uno non invertente, con tempi di propagazione di circa 90 ns. Infatti è circa di 90 ns il tempo massimo occorrente agli indirizzi per stabilizzarsi (vedi esempio n° 1, ricordando che i contatori sono due).

A.C. CHARACTERISTICS FOR 2101A-2 (250 ns ACCESS TIME)

READ CYCLE T_A = 0°C to 70°C, V_{CC} = 5V ±5%, unless otherwise specified.

Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{RC}	Read Cycle	250			ns	
t_A	Access Time			250	ns	t_r, t_f = 20ns
t_{CO}	Chip Enable To Output			180	ns	Input Levels = 0.8V or 2.0V
t_{OD}	Output Disable To Output			130	ns	Timing Reference = 1.5V
t_{DF}[3]	Data Output to High Z State	0		180	ns	Load = 1 TTL Gate
t_{OH}	Previous Read Data Valid after change of Address	40			ns	and C_L = 100pF.

WRITE CYCLE

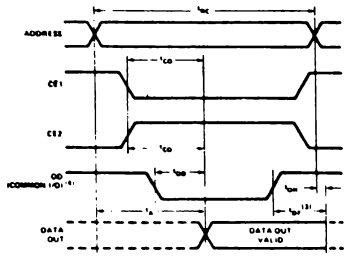
Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{WC}	Write Cycle	170			ns	
t_{AW}	Write Delay	20			ns	t_r, t_f = 20ns
t_{CW}	Chip Enable To Write	150			ns	Input Levels = 0.8V or 2.0V
t_{DW}	Data Setup	150			ns	Timing Reference = 1.5V
t_{DH}	Data Hold	0			ns	Load = 1 TTL Gate
t_{WP}	Write Pulse	150			ns	and C_L = 100pF.
t_{WR}	Write Recovery	0			ns	
t_{DS}	Output Disable Setup	20			ns	

CAPACITANCE[2] T_A = 25°C, f = 1MHz

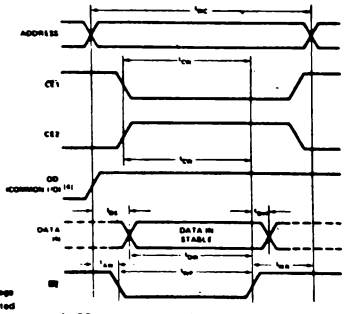
Symbol	Test	Limits (pF)	
		Typ.[1]	Max.
C_{IN}	Input Capacitance (All Input Pins) V_{IN} = 0V	4	8
C_{OUT}	Output Capacitance V_{OUT} = 0V	8	12

WAVEFORMS

READ CYCLE



WRITE CYCLE



NOTES 1 Typical values are for T_A = 25°C and nominal supply voltage
2 This parameter is periodically sampled and is not 100% tested
3 t_{DF} is with respect to the trailing edge of CE_1, CE_2, or OD, whichever occurs first
4 OD should be tied low for separate I/O operation.

FIG. 6. - Caratteristiche a.c. della RAM 2101-A (Intel).

WRITE CYCLE (Input pulse levels = 0.8 V to 2.0 V)

Characteristic	Symbol	MCM2112AL,P		MCM2112AL2,P2		MCM2112AL4,P4		Unit
		Min	Max	Min	Max	Min	Max	
Write Cycle Time	$t_{cyc(W)}$	350	–	250	–	450	–	ns
Chip Enable Pulse Width	t_{CE}	250	–	180	–	300	–	ns
Address Setup Time	t_{AS}	20	–	20	–	20	–	ns
Address Hold Time	t_{AH}	0	–	0	–	0	–	ns
Address to Write Release	t_{AWR}	350	–	250	–	450	–	ns
Write Pulse Width	t_W	250	–	180	–	300	–	ns
Data Setup Time	t_{DS}	150	–	100	–	190	–	ns
Data Hold Time	t_{DH}	0	–	0	–	0	–	ns

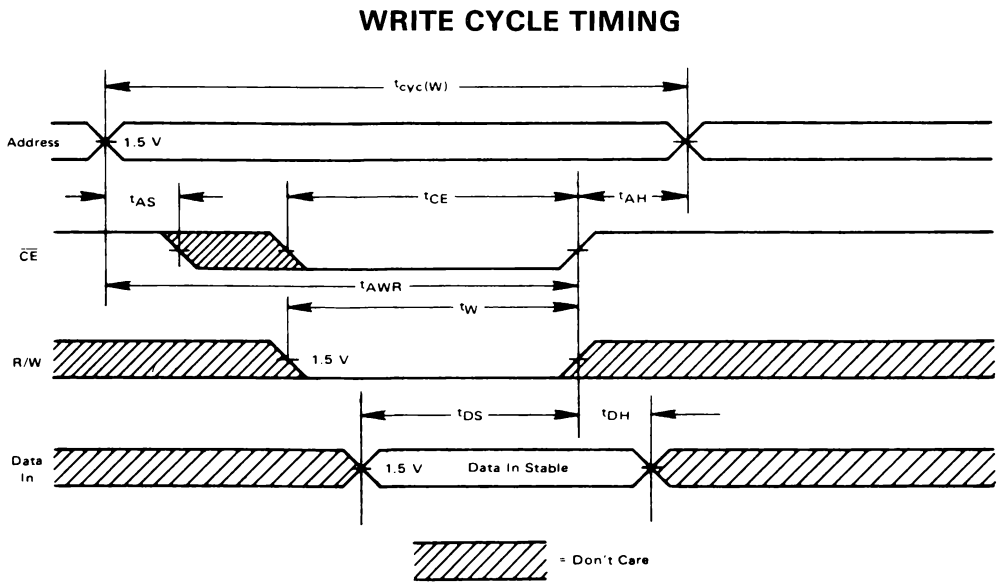


FIG. 7. - Caratteristiche a.c. della RAM 2112-A (Motorola).

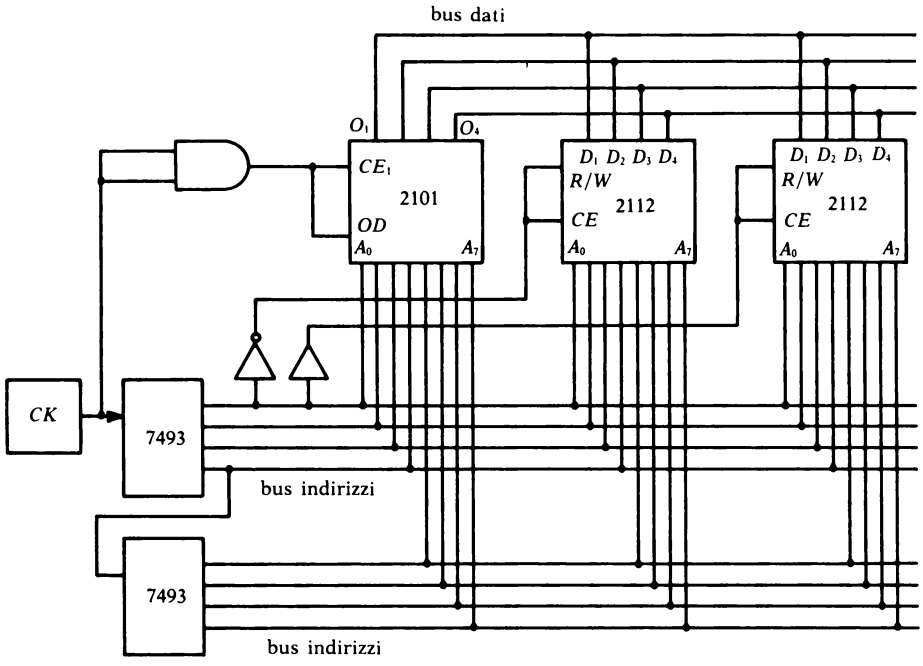


FIG. 8. - Trasferimento dalla RAM 2101 alle RAM 2112.

La figura 9 mostra il diagramma completo delle temporizzazioni.

L'abilitazione dell'elemento sorgente, che avviene tramite i comandi CE_1 ed OD, si può ottenere tramite porte AND C-MOS i cui tempi di propagazione, dell'ordine di 125 ns, realizzano il ritardo desiderato.

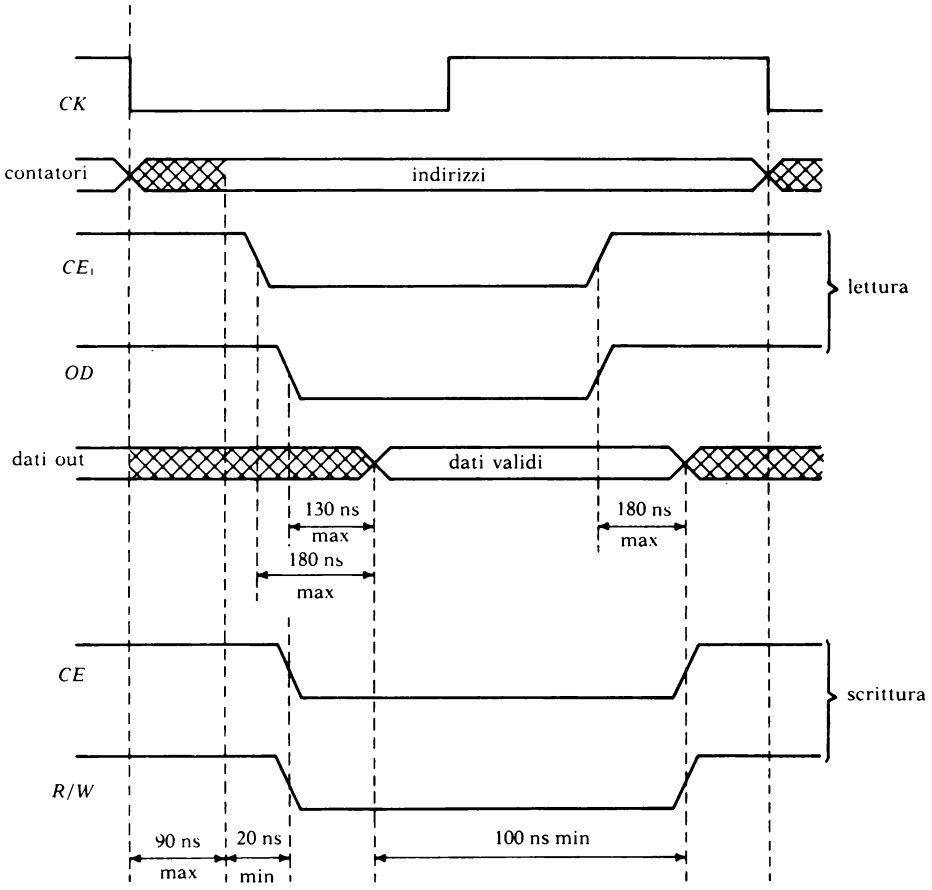


FIG. 9. - Tempi di propagazione per la lettura e scrittura.

L'esatto rispetto di tutte le temporizzazioni per le abilitazioni e disabilitazioni, diventa tanto più stringente quanto più elevata è la frequenza di lavoro che riduce i margini di tolleranza.

Nei progetti a più vasta scala, è l'uso intensivo dell'elaboratore che permette di controllare il rispetto di tutte le diverse esigenze, sollevando i progettisti da un tipo di esame assai difficoltoso.

c) trasferimento a collettore

Questo metodo è esattamente il contrario di quello a diffusione, realizzando il trasferimento di informazioni da più sorgenti ad un'unica destinazione in tempi successivi. Come è mostrato dalla figura 10, il bus è monodirezionale e sono necessarie linee per il controllo sia dell'elemento destinazione che per quelli sorgente. Infatti l'elemento D necessita del segnale di STB per sapere in quale istante è presente il dato in modo valido al suo ingresso, mentre gli elementi S devono essere abilitati, uno alla volta, a porre il dato sul bus.

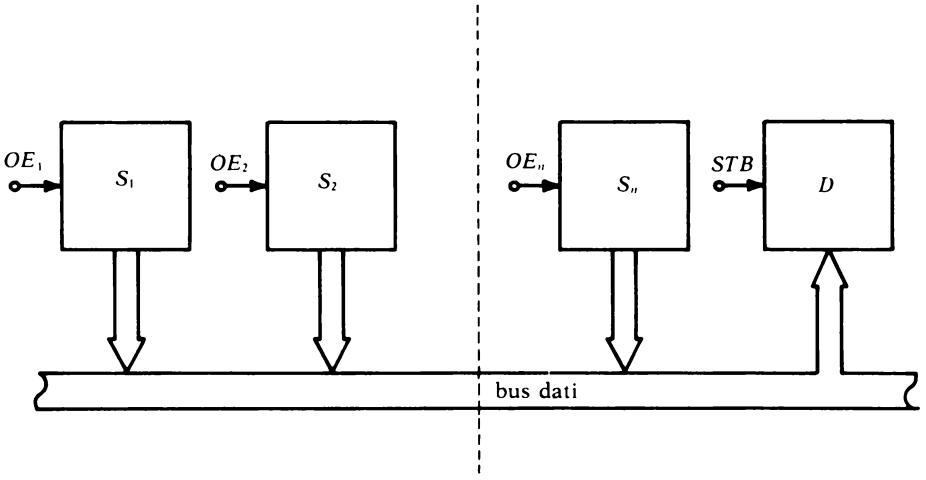


FIG. 10. - Trasferimento a collettore.

Ciò può essere effettuato tramite una serie di segnali di controllo di abilitazione OE (Out Enable). Inoltre i dispositivi Sorgente, poichè hanno le uscite collegate ad un'unica linea, devono essere di tipo *open-collector* o *tri-state,* onde evitare conflitti di competenza che potrebbero accadere se una uscita a livello basso fosse conessa ad un'altra uscita a livello alto.

Nella figura 11 è mostrato un esempio di sequenza temporale per il trasferimento di dati da S_3 a D, poi da S_1 a D ed infine da S_2 a D.

I tempi t_{en} (enable = abilitazione) e t_{dis} (disable = disabilitazione) sono rispettivamente i tempi necessari alle porte tri-state per abilitarsi o disabilitarsi su comando dei vari OE.

ESEMPIO 3.

Un esempio molto noto di trasferimento a collettore è dato dal *data logger* uno strumento adatto alla acquisizione di dati sia analogici che digita-

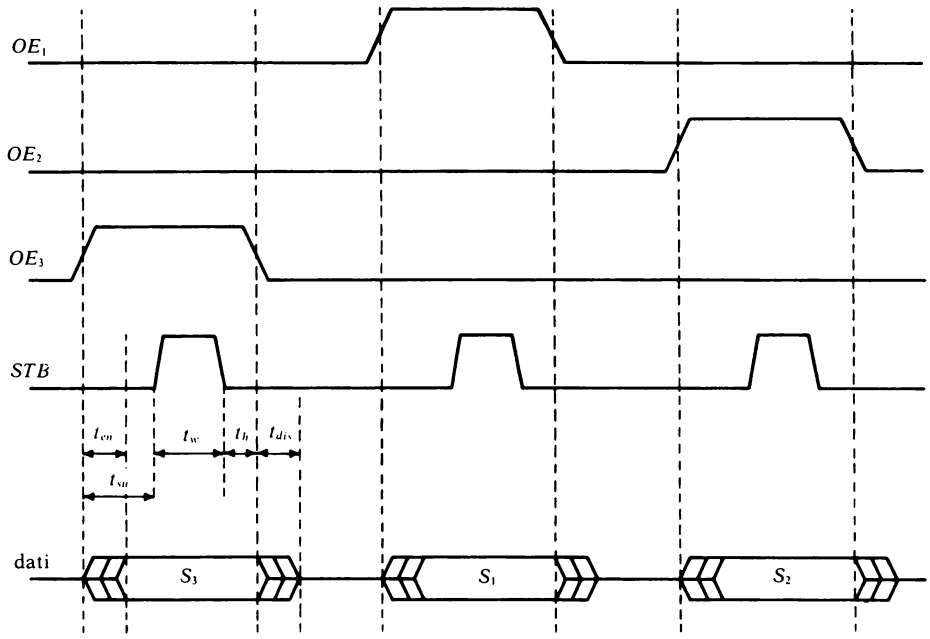


FIG. 11. - Segnali di controllo per un trasferimento a collettore.

li, provenienti da varie decine o centinaia di ingressi.

Il controllore è normalmente incorporato ed è in grado di svolgere anche una discreta varietà di funzioni diverse essendo programmabile, ma ove ci sia la necessità di elaborazioni più complesse, può essere interfacciato con un calcolatore più potente.

d) trasferimento a bus

Una connessione a bus realizza il trasferimento fra più elementi Sorgente a più elementi Destinazione, come è mostrato nella figura 12.

Anche in questo caso, i vari elementi Sorgente devono avere le uscite di tipo tri-state, oppure open-collector, poiché esse sono collegate al medesimo bus, che è ora bidirezionale.

L'elemento di controllo del sistema è generalmente esterno, dovendo gestire più segnali di *enable* e di *strobe*. Ad esempio volendo trasferire informazioni da S_2 a D_1 e poi da S_1 a D_n, la successione degli impulsi di controllo da inviare sarebbe OE_2 + STB_1 che realizza il primo trasferimento e quindi OE_1 + STB_n che realizza il secondo.

Il più delle volte però, le funzioni di sorgente e destinazione non sono così ben definite, potendo essere ogni elemento sia l'uno che l'altro. Ad

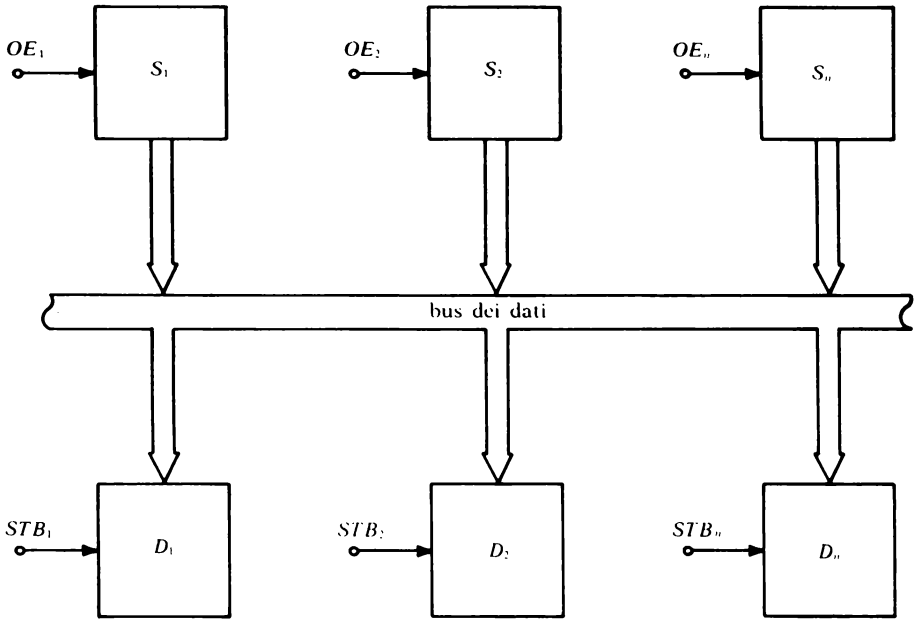


FIG. 12. - Trasferimento a bus.

esempio le memorie ROM sono elementi solo di tipo sorgente, mentre le RAM si comportano da sorgente nelle operazioni di lettura e da destinazione nelle operazioni di scrittura.

In un sistema di elaborazione dati, il controllo è realizzato dalla CPU o microprocessore che è al tempo stesso elemento sorgente e destinazione dovendo svolgere anche altri compiti dei quali si parlerà nel seguito. Esso è di gran lunga la parte più importante ed infatti occupa da solo circa il 70% dell'area di silicio costituente il μP stesso.

In figura 13 è mostrata una rappresentazione a bus di un sistema a microprocessore. Elemento caratterizzante di questa nuova configurazione è la presenza di tre distinti bus:

1) bus dei controlli; (monodirezionale, uscente dalla CPU) fornisce i segnali di abilitazione e di strobe specificando se un elemento è sorgente o destinazione oltre alle temporizzazioni per l'intero sistema. Nei micro ad 8 bit è composto da circa 5 linee, arrivando a più del doppio per quelli a 16 bit.
2) bus dei dati; (bidirezionale) trasporta le informazioni sulle quali operare e generalmente è costituito da 4, 8 o 16 linee.
3) bus degli indirizzi; (mono o bidirezionale) trasporta gli indirizzi che consentono di individuare l'elemento o gli elementi che partecipano in

un dato istante allo scambio di informazioni. Negli attuali microprocessori è ad 8 o 16 linee il che permette di indirizzare direttamente 2^{16} = 65.536 = 64 K di memoria in quest'ultimo caso.

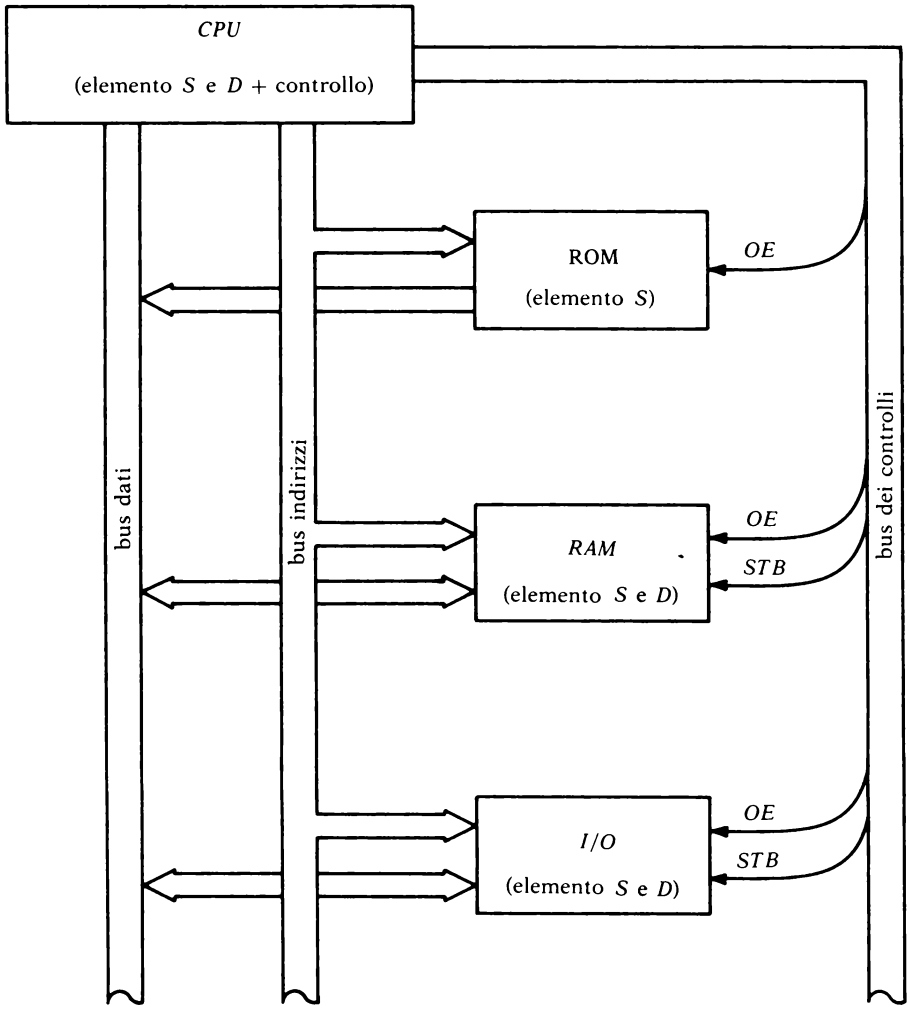


FIG. 13. - Schema a bus di un microcomputer.

2. Reti sequenziali come controllori

L'attivazione del trasferimento e dell'elaborazione di informazioni nei diversi punti di un sistema è il compito di una particolare sezione posta all'interno della CPU che prende il nome di «blocco di controllo». Questo particolare compito, immaginando l'intero sistema da gestire come composto da registri, sommatori, contatori, decodificatori, memorie e tutte le linee di trasferimento fra di essi, può essere assegnato a reti di tipo sequenziale impiegandone le uscite in una opportuna sequenza.

Il controllo di tali linee può essere in genere riconducibile a porte AND comandate, in istanti opportuni, da impulsi EN che abilitano il passaggio del segnale.

Nel seguito, a scopo esemplificativo, verranno usati schemi e notazioni semplificate come quelle mostrate nella seguente tabella 1. Si tenga presente che le operazioni indicate avvengono subito dopo il fronte di salita di EN senza tener conto di tutti i ritardi e tempi di propagazione.

Nell'ultimo caso mostrato nella tabella, alla memoria sono stati associati due registri specializzati, il MAR (Memory Address Register) e l'MDR (Memory Data Register). Il loro compito è di memorizzare temporaneamente gli indirizzi ed i dati rispettivamente che si presentano ad un certo istante alla memoria.

All'interno di una CPU, possono pensarsi come facenti parte della memoria stessa per cui nel seguito del testo, per semplicità, negli schemi a blocchi raffiguranti un μP non verranno più rappresentati.

Controllori autonomi

Un sistema che ripeta sempre le stesse operazioni elementari può essere controllato da una rete sequenziale autonoma. Infatti le reti autonome, non avendo ingressi I dall'esterno, hanno una evoluzione di tipo periodico e si può intervenire su di esse solo allo stato iniziale. Una volta avviato il sistema, esse ripetono all'infinito la sequenza prestabilita.

Come esempio si supponga di voler realizzare il controllo del sistema mostrato in figura 14, costituito da una memoria con 256 locazioni da 16 bit ciascuna (2 byte), un registro con funzioni da contatore (CON) ad 8 bit ed un registro di ingresso/uscita (I/0) anch'esso ad 8 bit.

I segnali EN_1, EN_2, EN_3, EN_4, ed EN_5, sono i segnali di controllo del sistema emessi dalla rete sequenziale (controllo), in una sequenza opportuna e sincronizzati da un clock principale.

Il funzionamento che si vuole ottenere è il seguente:
i dati provenienti dall'esterno, in gruppi di 8 bit, devono essere caricati nel registro I/0 sincronicamente con gli impulsi CK (ad ogni impulso di CK un byte viene memorizzato in I/0); dal registro I/0 i dati vengono allocati due a due nel registro MDR per costituire la parola di 16 bit da memorizzare.

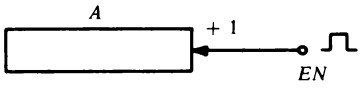
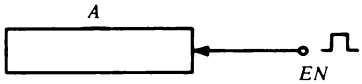
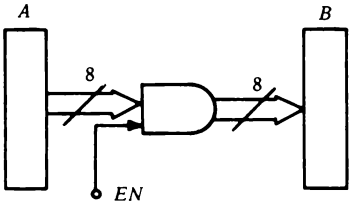
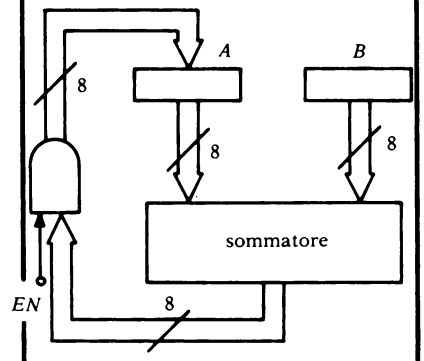
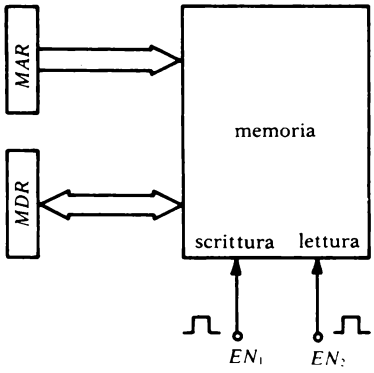
NOTAZIONE	SCHEMA	SIGNIFICATO
A+1 → A		Il contenuto di A si incrementa di 1 su comando di EN.
RSL (A) → A		Il contenuto di A è ruotato a destra, su comando di EN.
A → B		Trasferimento di 8 bit in parallelo tra i registri A e B, su comando di EN.
A+B → A		La somma dei contenuti dei registri A e B, va in A su comando di EN.
MDR → (MAR) (MAR) → MDR		EN_1: abilita a scrivere nella locazione specificata da MAR il contenuto del registro MDR. (scrittura). EN_2: abilita a trasferire il contenuto della locazione specificata dal MAR, nel registro MDR (lettura)

TABELLA 1.

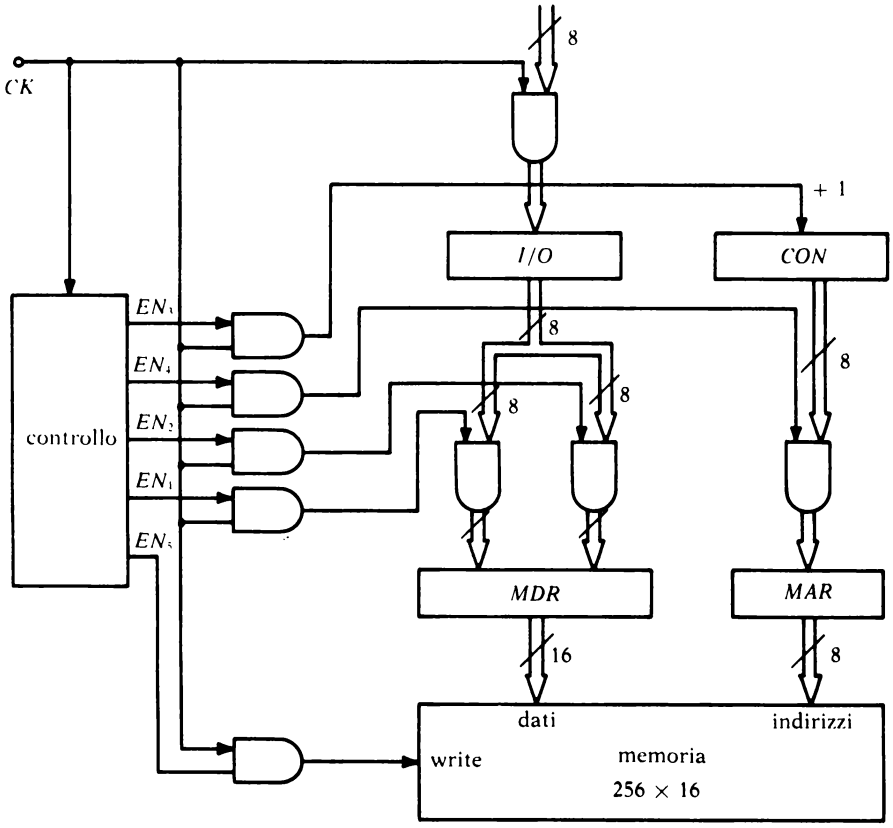


FIG. 14. - Scrittura in memoria di due byte contemporaneamente.

Il registro contatore CON fornisce al MAR l'indirizzo in cui va memorizzata la parola, incrementandosi di uno ogni due impulsi di clock.

Il controllore autonomo da progettare, deve essere mantenuto in uno stato X indifferente prima e dopo che è terminata l'operazione, mentre deve essere portato in uno stato Y iniziale dal quale avviare il ciclo composto dalle seguenti operazioni:

1) scrittura del dato nella memoria (EN_5 attivo)
2) I/0 → MDR, in MDR viene caricato il primo byte (EN_1 attivo)
3) CON + 1 → CON, il contatore si increm. di uno (EN_3 attivo)
4) I/0 → MDR, in MDR viene caricato il secondo byte (EN_2 attivo)
5) CON → MAR, si presenta un nuovo indirizzo (EN_4 attivo).

Si supponga che all'inizio sia il MAR che l'MDR contengano già i rispettivi byte.

È importante che tutta la precedente sequenza venga eseguita soltanto nell'arco di due impulsi di clock, poiché il registro I/0 viene continuamen-

te caricato dall'esterno ad ogni impulso per cui in caso contrario si perderebbero dei dati.

Le operazioni sopra indicate debbono perciò essere eseguite tutte nei due seguenti raggruppamenti;

1 + 2 + 3, segnali attivi EN_5, EN_1, EN_3
4 + 5, segnali attivi EN_2, EN_4

Si è ora in grado di disegnare il diagramma degli stati del controllore che è mostrato in figura 15, ove si è usata la convenzione di assegnare lo stato 0 ai segnali EN non attivi e lo stato 1 a quelli attivi.

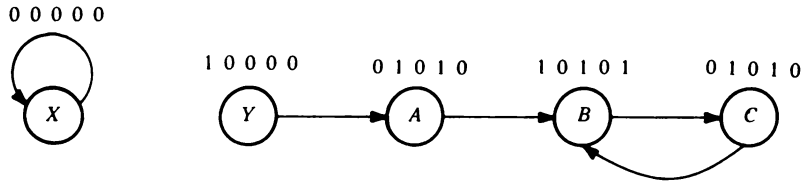


FIG. 15. - Diagramma degli stati del controllore

Quando il controllore è inattivo, si trova nello stato X generando segnali di EN sempre al livello zero, mentre invece viene portato in Y per iniziare il caricamento. Da qui, attraverso lo stato A arriva agli stati B e C che ripete autonomamente entrando in ciclo ed eseguendo tutte le operazioni in maniera ripetitiva.

Il suo arresto avviene solamente riportandolo forzatamente nello stato X.

Dal diagramma di flusso risulta infine agevole l'eventuale progetto della rete sequenziale autonoma che lo soddisfi.

Controllori con ingressi esterni

Per aumentare la flessibilità di impiego di sistemi sottopposti a controllori, si deve ricorrere all'utilizzazione di segnali provenienti dall'esterno che, in qualche modo, possono influenzarne almeno gli stati di partenza e di arrivo. A tale scopo ci si serve degli ingressi I di una rete sequenziale (vedi fig. 5, cap. 1), che potendo essere impiegati dall'esterno senza vincoli, costituiscono il mezzo per trasmettere opportune *istruzioni* alla rete stessa. Queste istruzioni, inviate in un codice di zeri e di uno nella loro forma più semplice, verranno eseguite dando origine ad una serie di operazioni comandate da *microistruzioni.*

In definitiva le microistruzioni sono costituite dai raggruppamenti dei segnali di abilitazione EN generati dalla rete di controllo.

Per chiarire quanto detto, si prenda ad esempio lo schema di figura 16 costituito da un elemento sommatore, un registro accumulatore A ed un

registro generico B. L'intero sistema opera alle dipendenze di una rete di controllo che accetta dall'esterno, sul suo registro IR (Instruction Register), le seguenti istruzioni:

codice	operazione	commenti
00	ALT	il sistema è fermo
01	A - B → A	al contenuto di A viene sottratto il contenuto di B
11	B - A → A	al contenuto di B viene sottratto il cont. di A che conterrà il risultato.

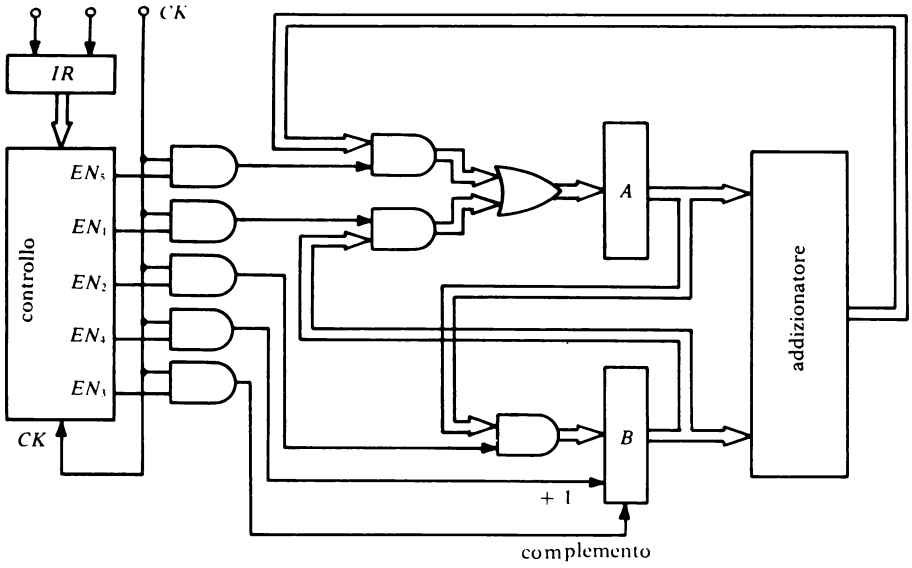


FIG. 16. - Controllore con ingressi esterni.

L'elemento sommatore realizza la sottrazione in complemento a due, vale a dire complementando il sottraendo ed addizionando i due dati.

Per semplicità si supponga anche in questo caso di trascurare i vari tempi di ritardo e di propagazione attraverso le porte ed i registri e che la complementazione venga effettuata solo nel registro B (il complemento a due si ottiene complementando ciascun bit ed aggiungendo 1 al risultato).

Per ciascuna delle tre istruzioni che possono essere inviate dall'esterno,

il controllo deve far eseguire una serie di microistruzioni inviando i segnali di abilitazione EN seguenti:

codice istruzione	microistruzioni	segnali del controllo				
		EN_1	EN_2	EN_3	EN_4	EN_5
IR = 00	ALT	0	0	0	0	0
IR = 01	$\overline{B} \rightarrow B$	0	0	1	0	0
	$B + 1 \rightarrow B$	0	0	0	1	0
	$A + B \rightarrow A$	0	0	0	0	1
IR = 11	$A \rightarrow B, B \rightarrow A$	1	1	0	0	0
	$\overline{B} \rightarrow B$	0	0	1	0	0
	$B + 1 \rightarrow B$	0	0	0	1	0
	$A + B \rightarrow A$	0	0	0	0	1

Realizzando lo schema di flusso, sarebbe a questo punto relativamente semplice progettare il controllore utilizzando gli usuali metodi di sintesi dei circuiti sequenziali.

Se nel registro delle istruzioni IR fosse caricato il codice 10, che non è stato previsto in fase di progetto, la decodifica della rete di controllo non la accetterebbe facendo rimanere il sistema nello stato di Alt.

Si comprende facilmente anche, che i codici delle istruzioni non debbano essere necessariamente espressi in binario, ma in una qualsiasi altra forma (ottale, esadecimale, ecc.) utile ai fini del linguaggio desiderato.

Controllori con ingressi di ritorno

Nel capitolo secondo, si è visto come, nella stesura dei programmi delle elaborazioni, si faccia comunemente ricorso ad informazioni che vengono generate nel corso dell'elaborazione stessa. È ciò che avviene ad esempio con le parole dei flag che possono considerarsi come lo specchio dello stato dell'elaborazione, oppure i segnali di ritorno da un sistema di controllo a retroazione.

Si prenda ad esempio un sistema come quello mostrato in figura 17, nel quale viene eseguita un'istruzione piuttosto che un'altra sulla base di un segnale proveniente dal suo interno.

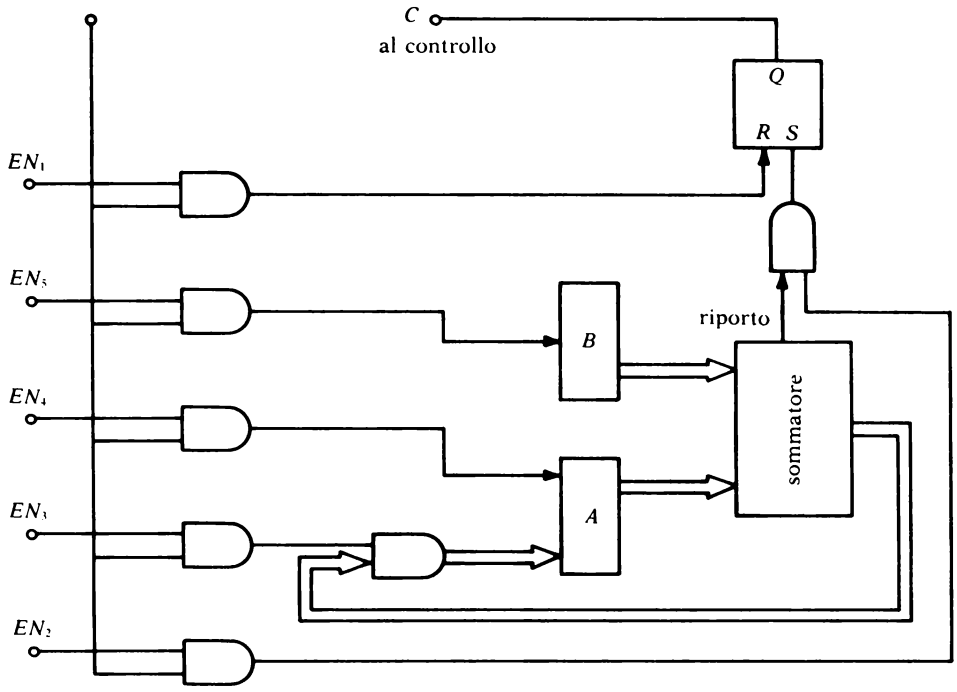


FIG. 17. - Controllore con un ingresso proveniente dal sistema.

Il registro istruzioni può essere caricato con uno dei codici operativi 00 oppure 11 che specificano le seguenti istruzioni:

IR = 00 PONE IN ALT IL SISTEMA
IR = 11 Se la somma dei contenuti di A e B non dà overflow, allora A + B va in A altrimenti A/2 + B/2 va in A.

Il flip-flop S-R è utilizzato per immagazzinare un segnale C proveniente dall'elemento sommatore, nel caso che l'addizione del contenuto dei due registri A e B generi un riporto.

È il segnale C che, proveniente dal sistema, indica alla rete di controllo di generare una diversa sequenza di microistruzioni per dividere per due il contenuto dei registri. Ricordando che la divisione per due si ottiene, in binario, facendo scorrere di un bit verso destra la parola, le microistruzioni ed i segnali generati dalla rete di controllo sono:

codice istruzione	microistruzioni	segnali del controllo				
		EN_1	EN_2	EN_3	EN_4	EN_5
IR = 00	ALT	0	0	0	0	0
IR = 11	SE C = 0 allora					
	A + B → A	0	0	1	0	0
	SE C = 1 allora					
	A/2 → A					
	B/2 → B					
	0 → C	1	0	0	1	1
	A + B → A	0	0	1	0	0

Per semplicità si è considerato il f-f inizialmente posto nella posizione C = 0.

Nei microprocessori si può immaginare che il codice operativo di ciascuna istruzione individui, all'interno della CPU, l'indirizzo di partenza del microprogramma relativo a quella particolare istruzione e realizzato dai costruttori durante il processo di fabbricazione.

3. Esecuzione di un programma

Nei paragrafi precedenti si è visto come la parte preponderante nella gestione di un sistema è lo scambio di informazioni e controlli fra i vari elementi che lo costituiscono.

Nel caso di un sistema a microprocessore, tale scambio avviene sia tra i vari blocchi interni della CPU (registri, ALU, ecc.), che con i vari dispositivi esterni quali le memorie ed i display.

Il modo di interazione fra di essi, durante l'esecuzione di un programma, verrà chiarito con l'aiuto di un esempio:

Si supponga che un microelaboratore debba eseguire il seguente programma immagazzinato in memoria a partire dalla locazione 0FF0: (tale indirizzo non ha alcun significato particolare ma è stato scelto a caso, così come per la notazione simbolica si è utilizzata quella dello Z 80).

indirizzo	notazione simbolica
0FF0	LD A, n
0FF1	05
0FF2	ADD A, (HL)
0FF3	ALT

Il programma è composto da tre istruzioni che danno origine alla seguente sequenza di operazioni:

1) carica il numero 05H (in esadecimale) nell'accumulatore,
2) somma al contenuto dell'accumulatore il contenuto della locazione di memoria indirizzata dalla coppia di registri HL. Si supponga che nella locazione specificata sia stato caricato in precedenza il numero 06H.

Nella figura 18 è mostrato lo schema generico di una CPU collegata, tramite bus, ad una memoria RAM esterna ed ad un dispositivo di Input/Output che potrebbe essere una tastiera per scrivere il programma.

L'esecuzione del programma ha inizio ponendo nel Program Counter l'indirizzo della locazione di memoria ove è posto il primo byte della prima istruzione da eseguire. Nell'esempio attuale, nel PC verrà posto 0FF0H.

Successivamente il microprocessore prosegue in un'alternanza di due fasi ben distinte: la fase di *fetch* (prelievo) e la fase di *execute* (esecuzione).

Durante la fase di fetch il μP acquisisce dalla memoria esterna l'istruzione che verrà poi eseguita nella fase di execute.

La prima è sempre la stessa per tutte le istruzioni, non dipendendo da esse, mentre nella seconda viene avviato il microprogramma che dà esecuzione alla particolare istruzione, per cui la durata di questa fase varia al variare della complessità dell'istruzione stessa.

Per il programma di somma dell'esempio mostrato, l'alternarsi delle fasi è la seguente:

prima istruzione: LD A, 05
fase di fetch (vedi fig. 19)

1) il contenuto del PC (OFFO) viene posto sul bus degli indirizzi che individua una locazione di memoria;
2) il contenuto della cella 0FF0 della memoria esterna, viene posto nel registro istruzioni IR (in IR viene posto il codice operativo di LDA,n = 3E);
3) contemporaneamente al passo 2, il PC viene incrementato di uno (0FF 0 + 1 = 0FF1).

I tre passi precedenti sono ovviamente stati eseguiti su comando del controllo così come quelli delle fasi successive.

fase di execute (vedi fig. 20).

1) il contenuto del PC (0FF1) è posto sul bus degli indirizzi;
2) il contenuto della cella indirizzata dal PC (0FF1) = 05 viene posto nell'accumulatore;
3) il contenuto del PC viene incrementato di uno.

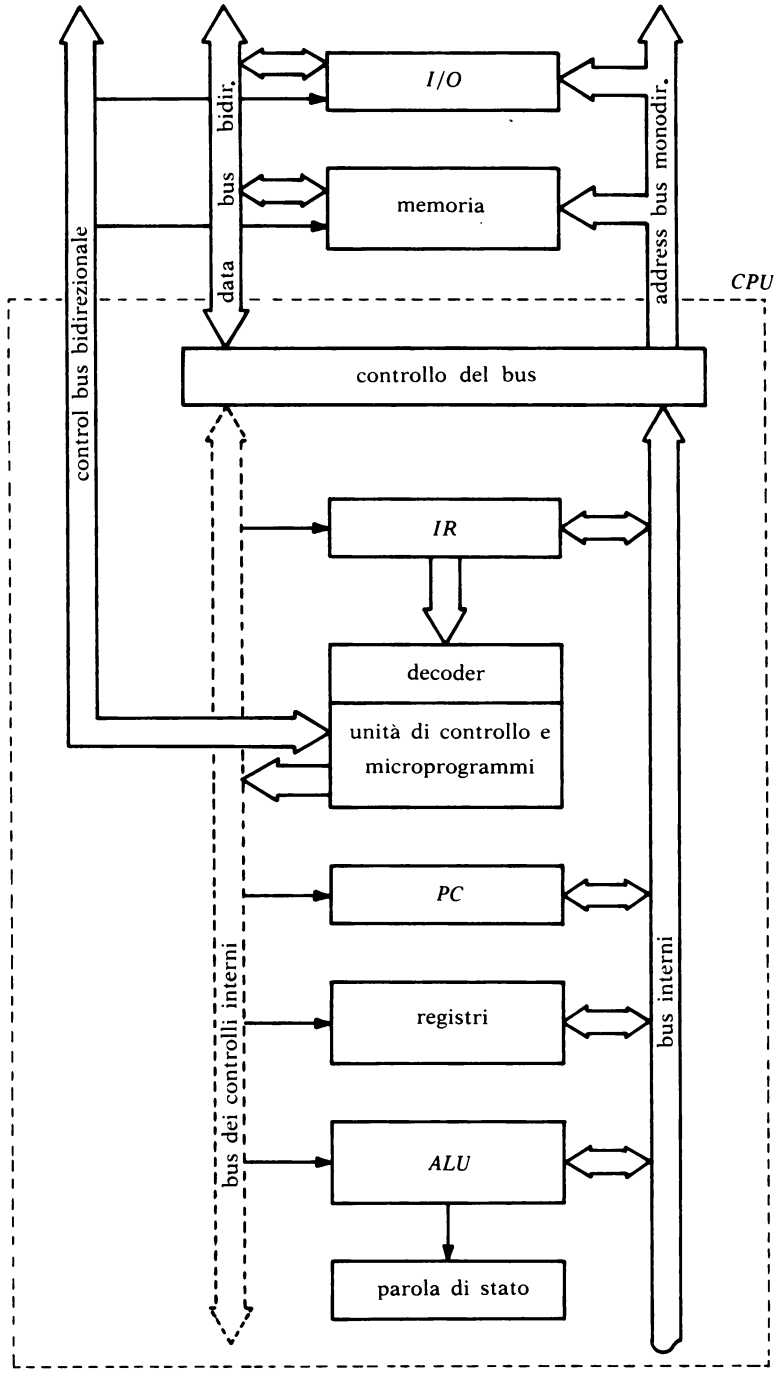


FIG. 18. - Microprocessore con elementi esterni.

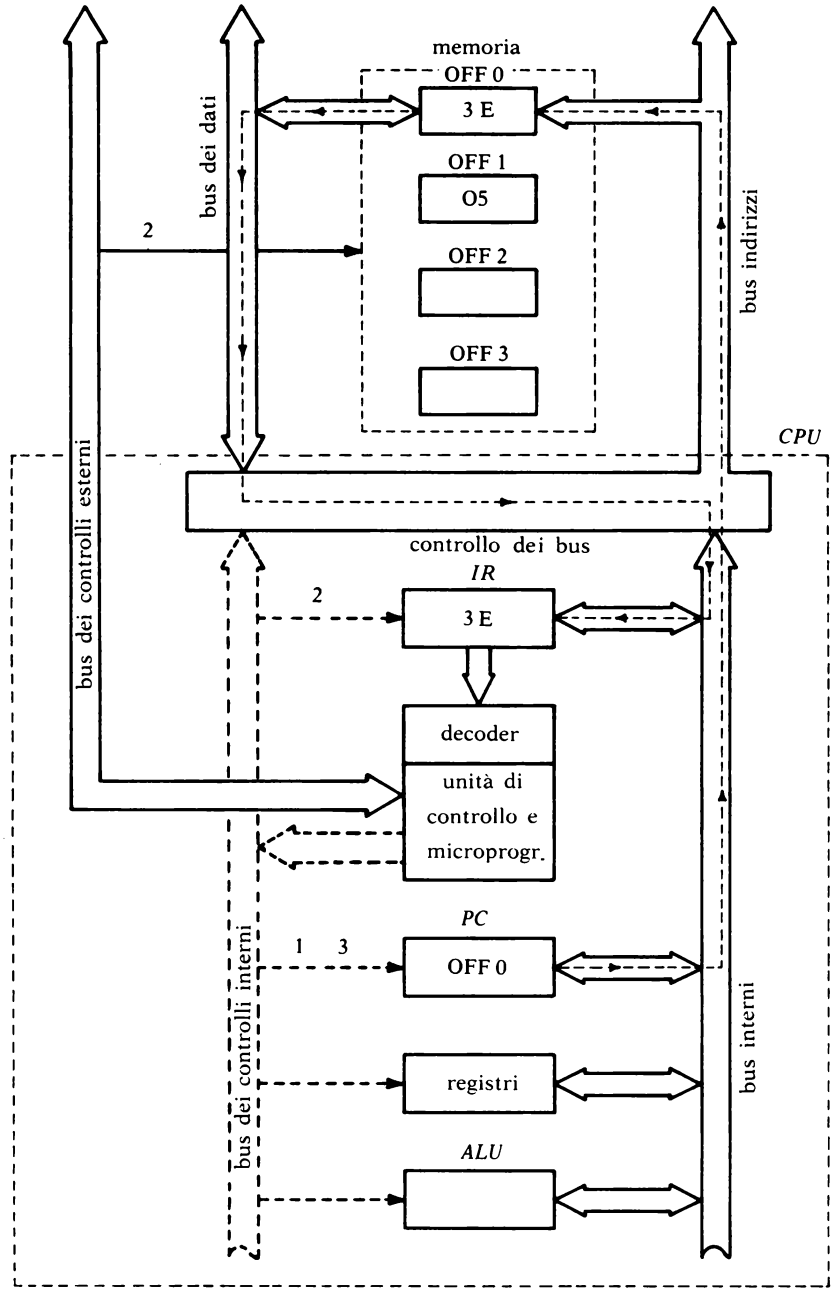


FIG. 19. - Fase di fetch.

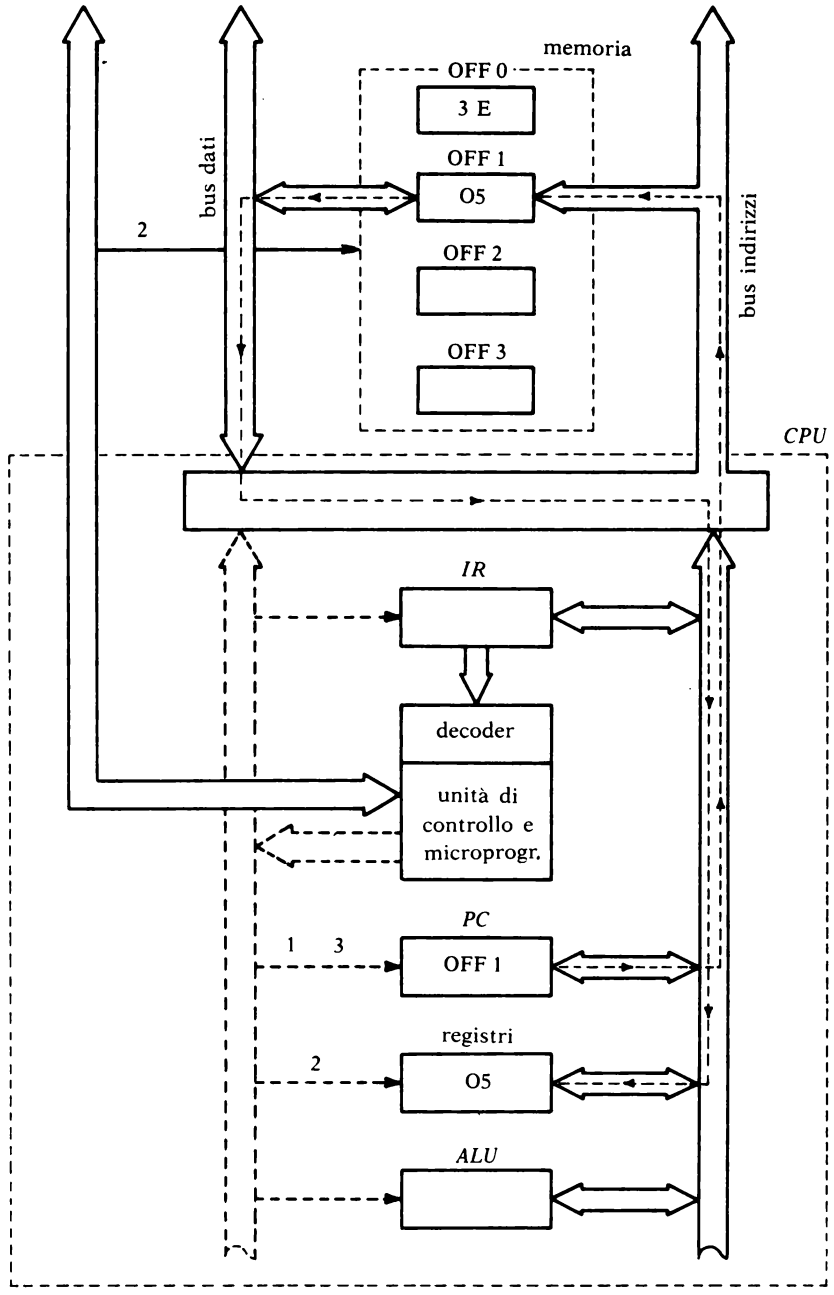


FIG. 20. - Fase di execute.

seconda istruzione: ADD A, (HL)

fase di fetch
1) il contenuto del PC (0FF2) viene posto sul bus degli indirizzi;
2) il contenuto della cella 0FF2 (86) viene posto nel registro istruzioni IR;
3) il PC viene incrementato di uno.

fase di execute

Inizia a questo punto l'esecuzione del microprogramma relativo alla istruzione ADD A, (HL), codificato con 86, il quale porta nella ALU (unità aritmetico logica) sia il contenuto dell'accumulatore che il contenuto della locazione di memoria indirizzata, ponendo il risultato nell'accumulatore stesso.

Come si è visto, un ruolo molto importante è svolto dal registro istruzioni IR. In esso viene posta l'istruzione letta dalla locazione di memoria indirizzata dal PC, e vi viene divisa in codice operativo ed indirizzo. È il codice operativo, diverso da istruzione ad istruzione, che dà origine alla serie di microistruzioni necessarie per l'esecuzione dell'istruzione vera e propria che prende anche il nome di «macroistruzione».

In alcuni casi, le istruzioni non contengono il dato su cui operare (come invece avveniva per LD A, 05) ma il suo indirizzo, ed in questi casi aumenta il numero di cicli per l'accesso in memoria. Comunque, è il numero totale di cicli macchina, il fattore più importante che influenza il tempo di esecuzione di un programma e questo è il modo in cui generalmente sono espressi i tempi di esecuzione delle istruzioni.

ESEMPIO

Viene presa in considerazione l'esecuzione di un programma che prevede il forzato cambiamento del contenuto del PC che avviene a causa di un salto condizionato, anziché il rispetto di un ordine regolarmente crescente.

Si supponga che il programma da eseguire sia il seguente:

indirizzo	codice mnem.	codice op.
0110	LD B, n	06
0111	n	AF
0112	DEC B	05
0113	JP NZ	C2
0114	L0	12
0115	HI	01
0116	HALT	76

Una volta caricato il Program Counter con l'indirizzo iniziale 0110, la successione delle operazioni per l'esecuzione del programma risulta essere:

fetch:
1) (PC) = 0110 viene posto sul bus indirizzi
2) (0110) = 06 viene posto nel registro istruzioni IR
3) Il PC viene incrementato di uno

execute:
1) (PC) = 0111 viene posto sul bus indirizzi
2) (0111) = AF viene caricato nel registro B
3) PC + 1 → PC

fetch:
1) (PC) = 0112 sul bus indirizzi
2) (0112) = 05 viene posto in IR
3) PC + 1 → PC

execute:
1) (B) = AF viene posto in ALU
2) ALU - 1 → ALU; il flag Z rimane a zero, ed il PC non viene incrementato di uno.

fetch:
1) (PC) = 0113 viene posto sul bus indirizzi
2) (0113) = C2 viene posto in IR
3) PC + 1 → PC

execute:
1) poiché il controllo rileva che Z = 0, il contenuto delle due caselle di memoria successive alla 0113 (dove c'era il codice operativo C2) viene interpretato come un indirizzo a due byte.
2) (0114) + (0115) → PC; il contenuto delle due locazioni di memoria viene caricato nel PC che conterrà così 0112.

Le alternanze precedenti delle fasi di prelevamento ed esecuzione proseguono nel modo precedente fino a quando il flag Z diverrà 1 perché si è azzerato il registro B. In questo caso non viene alterato il contenuto del PC e si continua con l'esecuzione dell'istruzione posta nella locazione 0116, in questo caso l'alt.

4. Le temporizzazioni e le linee di controllo

Le sequenze descritte nel paragrafo precedente, per ovvi criteri di chiarezza, sono state quasi del tutto svincolate da considerazioni circa il rispetto delle temporizzazioni e l'invio dei segnali di controllo.

È la CPU che, insieme al clock del sistema, fornisce i comandi necessari per indicare sia il tipo di operazione che l'istante in cui può aver luogo.

Nella pratica i 40 piedini di una CPU ad 8 bit, differiscono sia per il nome dato dai diversi fabbricanti che per la funzione svolta, ma in generale le operazioni fondamentali per l'esecuzione di una istruzione generica sono le seguenti:

— lettura o scrittura in memoria
— lettura o scrittura in un dispositivo esterno di I/0
— riconoscimento di una interruzione.

Ciascuna di queste costituisce un *ciclo macchina* che a sua volta è composto da più cicli di clock detti *cicli T.*

La durata dei cicli di clock dipende dalla sua frequenza; ad esempio per un clock da 4 MHz un ciclo dura 0,25 μs.

Nella figura 21 sono mostrate le temporizzazioni dello Z 80 per l'esecuzione della prima istruzione del programma precedente.

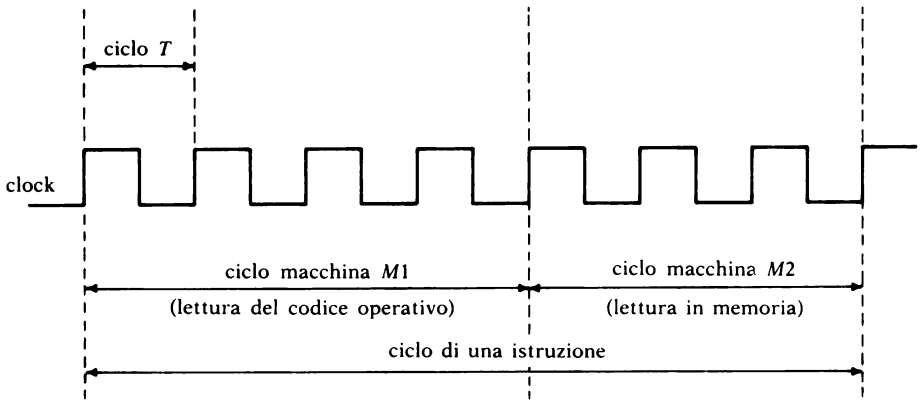


FIG. 21. - Esempio di ciclo per una istruzione.

Il primo ciclo macchina (M1), costituito da quattro cicli T nel presente esempio, costituisce la fase di fetch, che preleva dalla memoria l'istruzione per depositarla nel registro IR. Il successivo (o i successivi cicli macchina per istruzioni diverse), realizza l'esecuzione dell'istruzione scambiando i dati fra i vari blocchi del sistema.

Come già detto, l'esecuzione di scambi di dati ed informazioni, richiede da parte della CPU l'attivazione dei segnali di *strobe* e di *enable,* caratteristici di ciascun tipo di operazione elementare.

Nel seguito, a titolo di esempio, vengono descritte nel dettaglio le reali temporizzazioni ed i segnali di controllo impegnati dalla CPU Z 80 della

ZILOG (costruita anche dalla SGS italiana), per l'esecuzione dei vari cicli macchina.

Compito dei segnali di controllo è di autorizzare ad un'operazione solo il dispositivo interessato, per cui possono essere così raggruppati:

— per la lettura di un codice operativo sono interessati $\overline{MREQ}$, $\overline{RD}$, $\overline{M1}$, $\overline{RFSH}$, $\overline{WAIT}$.
— per la lettura o scrittura in memoria: $\overline{MREQ}$, $\overline{RD}$, $\overline{WR}$, $\overline{WAIT}$.
— per la lettura o scrittura in dispositivi di I/O: $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$, $\overline{WAIT}$.
— per la richiesta o riconoscimento del bus: $\overline{BSRQ}$, $\overline{BUSAK}$.
— per la richiesta o riconoscimento di interruzione: $\overline{INT}$, $\overline{M1}$, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, $\overline{WAIT}$.

Lettura di codice operativo

Nella figura 22 è presentato un ciclo M1 necessario per la lettura di un codice operativo dalla memoria, che nel caso dello Z80 può essere composto da 4, 5 o 6 cicli T, secondo la complessità della funzione da svolgere.

All'operazione sono interessati il bus dei dati ($D_0 \div D_7$), il bus degli indirizzi ($A_0 \div A_{15}$) ed i segnali di controllo $\overline{MREQ}$, $\overline{RD}$, $\overline{M1}$, $\overline{RFSH}$ ed eventualmente $\overline{WAIT}$.

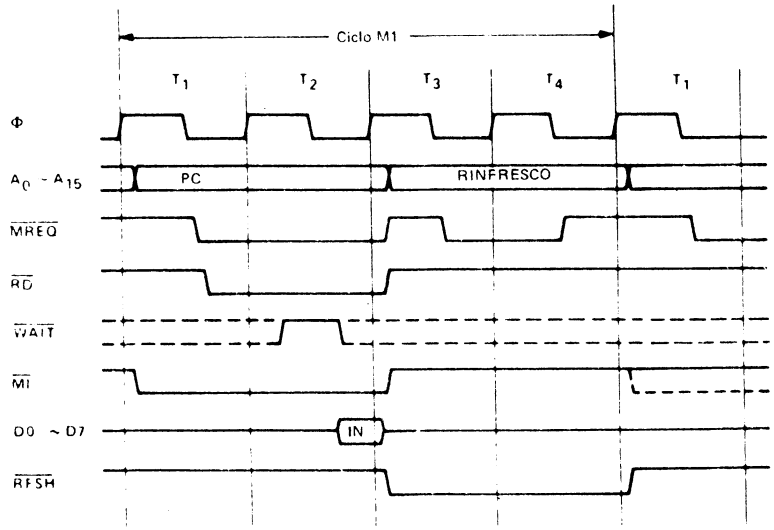


FIG. 22. - Ciclo di lettura di un codice operativo (SGS)

All'inizio del ciclo M1, il contenuto del PC, che è il primo byte di un'istruzione, viene posto sul bus degli indirizzi e contemporaneamente il segnale M1 diventa basso (attivo) per indicare che si sta leggendo un codice operativo.

Se quest'ultimo fosse a due byte, verrebbero eseguiti due cicli M1.

Dopo mezzo periodo di clock, per dar modo agli indirizzi di stabilizzarsi, diventa attivo il segnale $\overline{MREQ}$, che serve ad abilitare la memoria esterna, ed il segnale $\overline{RD}$ che fa depositare sul bus dati il contenuto della cella indirizzata.

A causa dei tempi di accesso e di abilitazione, il contenuto della memoria viene posto sul bus dopo questo tempo, per cui la CPU è forzata a leggere il dato in corrispondenza del fronte in salita del clock dello stato T_3, e contemporaneamente vengono disattivati i segnali $\overline{MREQ}$, $\overline{RD}$ ed $\overline{M1}$ (vengono portati al livello alto).

Volendo utilizzare frequenze molto elevate, si comprende come le memorie debbano avere necessariamente un tempo di accesso complessivo inferiore alla durata di un ciclo di clock.

I periodi T_3 e T_4, *poiché sono impiegati dalla CPU per la decodifica del codice operativo,* sono utilizzati per rinfrescare le eventuali memorie dinamiche, presentando sulla parte bassa del bus indirizzi ($A_0 \div A_7$) un indirizzo di memoria e facendo diventare attivo il segnale di $\overline{RFSH}$ insieme a $\overline{MREQ}$. L'operazione di refresh avviene eventualmente solo durante i cicli M1.

Il segnale di $\overline{WAIT}$, in ingresso alla CPU, che viene letto sul fronte in discesa di T_2, è utilizzato nei casi in cui un dispositivo esterno richieda di allargare il ciclo M1. Tale allargamento consiste di un ciclo T ogni volta che il fronte in discesa di T_2 trova il segnale $\overline{WAIT}$ basso; se al fronte successivo in discesa, $\overline{WAIT}$ è ancora basso, viene inserito un altro ciclo di attesa e così via fino a quando il segnale non viene disattivato.

Lettura / scrittura in memoria

Durante ciascuno di questi cicli tra la CPU ed il periferico, che nel caso particolare è una memoria esterna, occorrono i seguenti segnali per effettuare lo scambio di informazioni desiderato:

— una linea per autorizzare il periferico o la memoria a riconoscere il proprio indirizzo quando è presente in modo stabile sul bus;
— una linea per comandare l'operazione di lettura (porre il contenuto della locazione indirizzata, sul bus dei dati);
— una linea per comandare l'operazione di scrittura (porre il dato presente sul bus dati, nella locazione indirizzata).

Nel caso dello Z 80 se il periferico è una memoria, diventa attivo (basso) il segnale $\overline{MREQ}$, mentre se è un diverso tipo di periferico diventa attivo il segnale $\overline{IORQ}$. In questo modo ad un unico indirizzo corrisponde o una locazione di memoria o l'indirizzo di un altro dispositivo e sarà solo l'attivazione di uno dei due precedenti segnali, da parte della CPU, ad abilitare l'uno o l'altro.

Nella figura 23 è mostrato il diagramma temporale dei segnali coinvolti in un ciclo di lettura o scrittura.

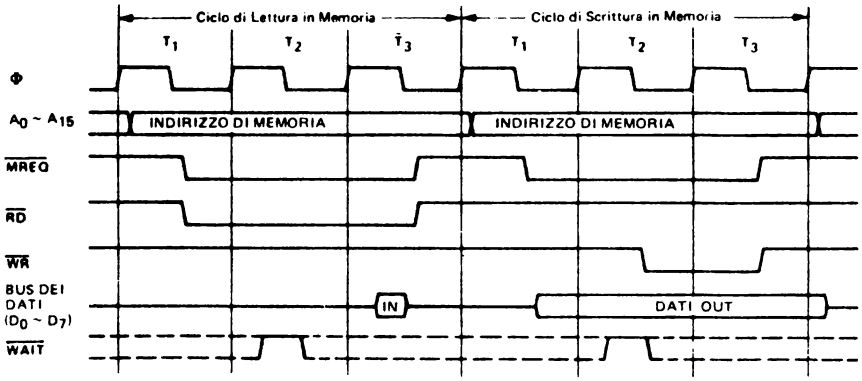


FIG. 23. - Ciclo di lettura o scrittura Z80 (SGS).

Si tenga presente che questi cicli non sono relativi alle fasi di lettura di un codice operativo per i quali è valido solo il ciclo M1 mostrato nella figura 22.

Anche in questo caso però, è necessario che i segnali di controllo diventino attivi dopo che le informazioni presenti sui bus dati ed indirizzi si sono stabilizzati e disattivi prima che cambino nuovamente. Ad esempio per un ciclo di lettura della memoria, il dato è disponibile sul bus dopo che sono trascorsi i tempi di accesso rispetto ad $\overline{MREQ}$ ed il tempo di abilitazione dei buffer a tre stati comandati da $\overline{RD}$. Dopo di ciò i registri interni della CPU possono leggerlo su comando di un segnale STB interno che deve rispettare i tempi t_{su} e t_h.

La funzione del comando di WAIT è identica a quella descritta per la fase di fetch e cioè quella di allungare i cicli di lettura o scrittura.

Cicli di ingresso/uscita

Come già illustrato nella precedente sezione, la sola differenza tra un ciclo di lettura o scrittura da una periferica o da una memoria, è l'utilizzo di un segnale dedicato al particolare compito.

Come è mostrato nella figura 24, si nota che viene inserito automaticamente un ciclo di attesa (WAIT).

Il ciclo di attesa viene inserito automaticamente perché spesso le periferiche sono più lente del microprocessore per cui durante la trasmissione o ricezione dei dati, inviano una richiesta di wait, ma in tal caso il tempo intercorrente tra quando il segnale $\overline{IORQ}$, diventa attivo a quando viene letta la linea di $\overline{WAIT}$ sarebbe stato troppo breve per il periferico per riconoscere il proprio indirizzo ed attivare la linea di attesa.

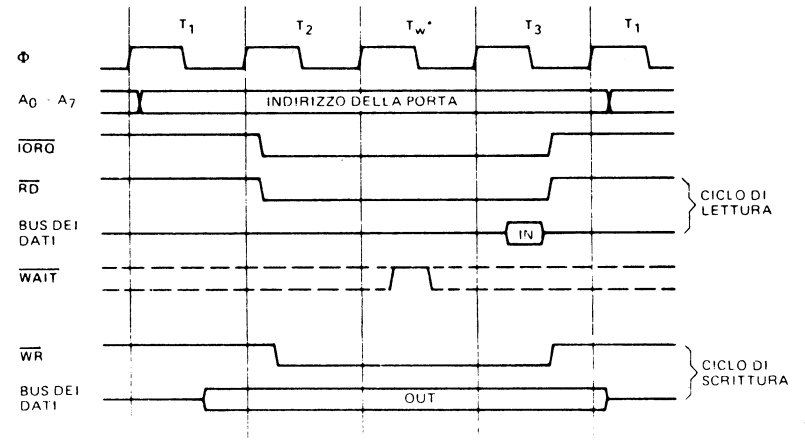


FIG. 24. - Ciclo di lettura o scrittura in un dispositivo periferico (Manuale tecnico SGS).

Richiesta del bus

In alcune operazioni è utile disattivare la CPU se essa non è chiamata direttamente ad operare.

Un esempio è dato dall'operazione di accesso diretto alla memoria (DMA = Direct Memory Access), nella quale avviene un veloce scambio di dati tra la memoria ed un dispositivo periferico senza passare per la CPU che provocherebbe dei rallentamenti.

Il periferico che richiede il DMA, deve generare un segnale da inviare all'unità centrale che, una volta riconosciutolo, si mette con tutte le sue uscite in tri-state (bus dati, bus indirizzi e controlli nello stato di alta impedenza).

In questo modo il μP è virtualmente sconnesso dai bus che vengono così utilizzati solo dai dispositivi esterni.

Nel caso dello Z 80, come mostra la figura 25, i segnali coinvolti sono

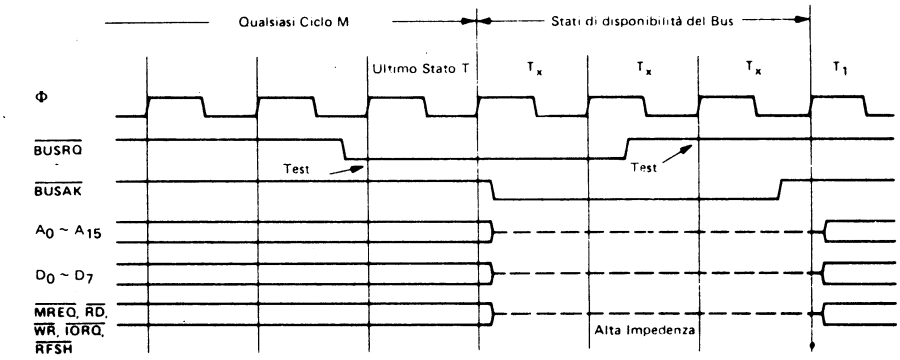


FIG. 25. - Ciclo di richiesta e riconoscimento del bus. (Manuale tecnico SGS).

$\overline{\text{BUSRQ}}$ (bus request) inviato dalla periferica e $\overline{\text{BUSAK}}$ (bus acknowledge) inviato dalla CPU al dispositivo richiedente per indicargli che si è posta in tri-state.

Alla fine di ogni ciclo macchina, vale a dire alla fine di ciascuna operazione elementare, in corrispondenza del fronte di salita dell'ultimo ciclo di clock, la CPU controlla lo stato del suo ingresso $\overline{\text{BUSRQ}}$ e se lo trova attivo (basso), si pone nello stato di alta impedenza, emettendo contemporaneamente il segnale $\overline{\text{BUSAK}}$ (basso).

I due punti di test indicati mostrano come la richiesta del controllo del bus venga soddisfatta solo sul fronte di salita iniziale dell'ultimo ciclo T e che lo stato di alta impedenza continua per un ciclo T dopo il riconoscimento della disattivazione di $\overline{\text{BUSRQ}}$.

Richiesta di interruzione

Un fattore molto importante e caratterizzante dei μP è la facoltà di ricevere segnali di interruzione inviati da dispositivi periferici, in seguito ai quali viene sospeso il programma in esecuzione per dar corso ad una particolare procedura legata al tipo di periferico che richiede l'interruzione stessa.

Nel capitolo 4° verrà svolto più dettagliamente questo argomento, limitando ora la descrizione alle temporizzazioni di un ciclo di richiesta e riconoscimento dell'interruzione da parte della CPU. Il ciclo è mostrato nella figura 26.

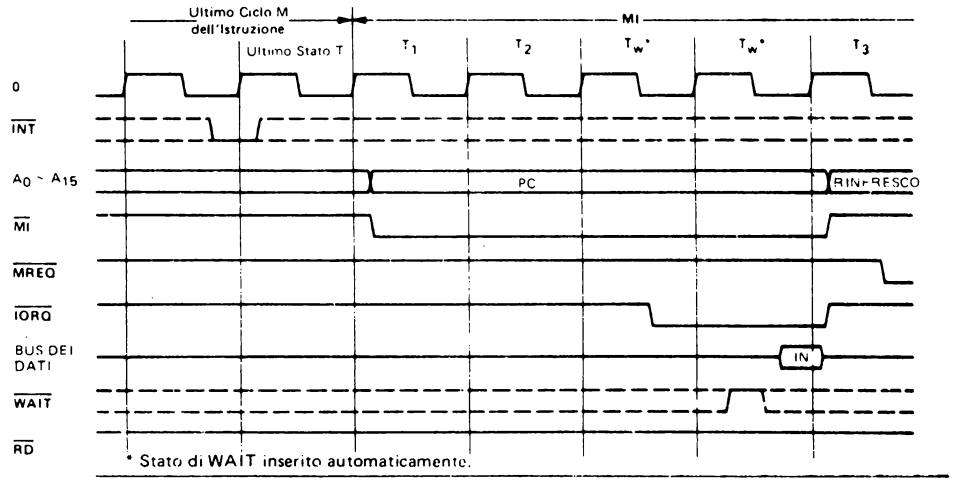


FIG. 26. - Ciclo di richiesta di interruzione. (Manuale tecnico SGS).

Il periferico, per richiedere l'interruzione, deve mandare al livello basso il segnale $\overline{INT}$. La CPU campiona ciclicamente il piedino INT, in corrispondenza del fronte di salita all'inizio dello ultimo stato T di un ciclo M, ed accetta il segnale solo se è stato settato ad 1, mediante il programma software, un f-f interno di abilitazione, o non è attivo $\overline{BUSRQ}$.

Una volta accettato il segnale, inizia un ciclo M1 particolare durante il quale viene attivata la periferica tramite $\overline{IORQ}$ e vengono inseriti automaticamente due stati di attesa T_W. Questo perché la periferica che ha richiesto l'interruzione abbia il tempo di farsi riconoscere inviando sul bus dati il proprio indirizzo.

5. Reset

L'ingresso per il segnale di controllo $\overline{RESET}$, presente in tutti i microprocessori, viene utilizzato per iniziare l'esecuzione di un programma, o per inizializzare la CPU dopo la sua accensione. Normalmente tale operazione consiste nel forzare il PC a contenere un determinato indirizzo, che di solito è lo zero, o una particolare locazione di restart da dove inizia il programma monitor.

Normalmente è prevista una attivazione automatica di questo ingresso all'atto dell'accensione, mediante un circuito R-C collegato direttamente all'alimentazione, come mostrato nella figura 27, nel caso che il segnale sia attivo al livello basso.

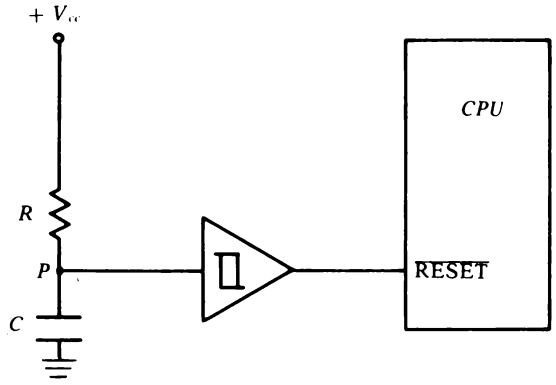


FIG. 27. - Reset automatico all'accensione.

Per dar modo alla CPU di completare tutte le operazioni di inizializzazione, la durata del comando di reset deve essere almeno di 3 ÷ 6 cicli T.

Per il calcolo dei valori della resistenza e del condensatore, viene ritenu-

ta trascurabile la corrente $I_{IH} = 40\,\mu A$, assorbita dal trigger di Schmitt. Infatti, il valore iniziale della corrente di carica del condensatore viene scelto almeno 10 volte maggiore di I_{IH}, dimensionando opportunamente la resistenza R.

Considerando il condensatore inizialmente scarico, la forma d'onda della tensione nel punto P, allorché viene alimentato il sistema, è quella mostrata in figura 28.

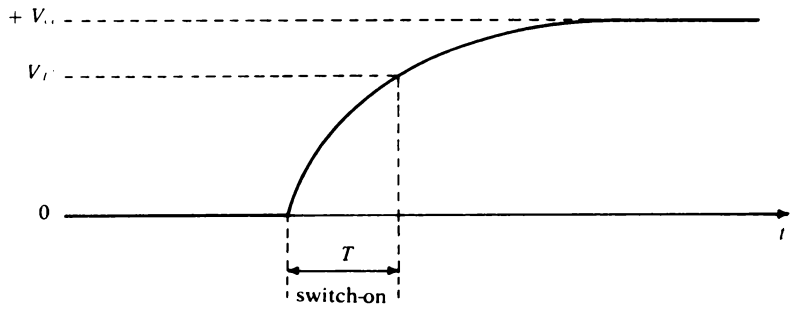


FIG. 28. - Forma d'onda all'ingresso di reset durante l'accensione.

Considerando che la tensione nel punto P debba rimanere al di sotto del valore della V_T, del trigger di Schmitt, per un tempo pari a T, si può scrivere:

$$v_P = V_{cc}\,(1 - e^{-\frac{T}{RC}}) \qquad [1]$$

da cui si ottiene:

$$1 - \frac{V_p}{V_{cc}} = e^{-\frac{T}{RC}}$$

ed infine:

$$RC = \frac{-T}{\ln\left(1 - \frac{V_p}{V_{cc}}\right)}$$

Scegliendo un valore di resistenza che soddisfi la condizione di essere percorsa da almeno dieci volte la I_{IH} si può calcolare in ultimo il valore della capacità del condensatore.

ESEMPIO

Considerando un circuito alimentato a + 5 V, una frequenza di lavoro di 1 MHz, una V_{T+} del trigger di Schmitt pari a 1,7 V e scegliendo un tempo T molto lungo (ad esempio 100 ms) per tener conto del ritardo dell'alimentatore a raggiungere la tensione nominale, si può scrivere:

$$1{,}7 = 5\left(1 - e^{-\frac{100 \cdot 10^{-3}}{RC}}\right)$$

da cui:

$$RC = \frac{-100 \cdot 10^{-3}}{\ln\left(1 - \frac{1{,}7}{5}\right)} = 240 \text{ ms.}$$

Scegliendo un valore di resistenza pari a 10 K Ω, si ottiene per la capacità:

$$C = \frac{0{,}24}{10 \cdot 10^{3}} = 24\ \mu F$$

Nei calcoli si è trascurata la resistenza interna del generatore di tensione, che di solito è di qualche decina di ohm.

Il comando di reset può essere dato anche manualmente, ed in questo caso uno di circuiti utilizzati è mostrato in figura 29, considerando che un operatore umano ben difficilmente riesce a tenere premuto un tasto per un tempo inferiore a 70 ÷ 80 ms, per cui è senz'altro garantito il tempo minimo di reset per qualsiasi μP.

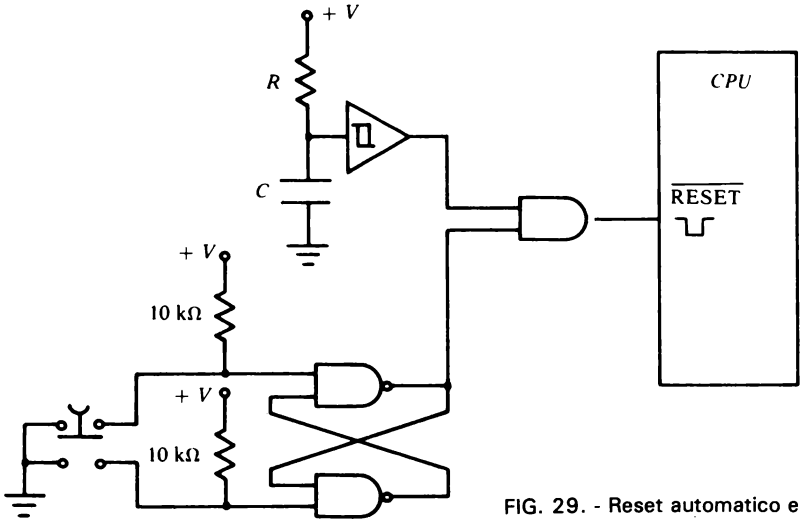


FIG. 29. - Reset automatico e manuale.

6. Caratteristiche elettriche

Nelle seguenti tabelle sono riportate, come esempio, le caratteristiche statiche e dinamiche della CPU Z 80, fornite dalla SGS.

VALORI MASSIMI ASSOLUTI*

Temperature under bias (unless otherwise specified)	0 to 70	°C
Storage temperature	-65 to +150	°C
Voltage on any pin with respect to ground	-0.3 to +7	V
Power dissipation	1.5	W

* Comment – Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

CARATTERISTICHE STATICHE (T_{amb} = da 0°C a 70°C, V_{CC} = 5V ± 5% se non diversamente specificato)

	Parametro	Condizioni di misura	Min.	Tip.	Max.	Unità
V_{ILC}	Clock input low voltage		-0.3		0.45	V
V_{IHC}	Clock input high voltage		V_{CC}-0.6		V_{CC}+0.3	V
V_{IL}	Input low voltage		-0.3		0.8	V
V_{IH}	Input high voltage		2		V_{CC}	V
V_{OL}	Output low voltage	I_{OL} = 1.8 mA			0.4	V
V_{OH}	Output high voltage	I_{OH} = -250 µA	2.4			V
I_{CC}	Power supply current for Z80 for Z80A			 90	150 200	mA mA
I_{LI}	Input leakage current	V_{IN} = 0 to V_{CC}			10	µA
I_{LOH}	Tri-state output leakage current in float	V_{OUT} = 2.4 to V_{CC}			10	µA
I_{LOL}	Tri-state output leakage current in float	V_{OUT} = 0.4V			-10	µA
I_{LD}	Data bus leakage current in input mode	$0 \leq V_{IN} \leq V_{CC}$			±10	µA

CAPACITA' (T_{amb} = 25°C, f = 1 MHz)

	Parametro	Condizioni di misura	Min.	Tip.	Max.	Unità
C_ϕ	Clock capacitance	unmeasured pins returned to ground			35	pF
C_{IN}	Input capacitance				5	pF
C_{OUT}	Output capacitance				10	pF

CIRCUITO DI MISURA

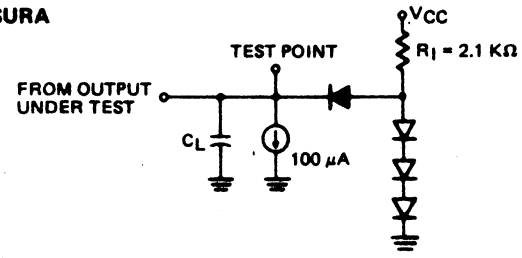


Tabella 2 (SGS)

CARATTERISTICHE DINAMICHE (T_{amb} = da 0°C a 70°C, V_{CC} = +5V ± 5% , se non diversamente specificato)

Z80A CPU

Segnale	Simbolo	Parametro	Condizioni di misura	Min.	Tip.	Max.	Unità
Φ	t_c	Clock period		0.25		[12]	μsec
	$t_{w(\Phi H)}$	Clock pulse width, clock high		110		[E]	nsec
	$t_{w(\Phi L)}$	Clock pulse width, clock low		110		2000	nsec
	$t_{r,f}$	Clock rise and fall time				30	nsec
A_{0-15}	$t_{D(AD)}$	Address output delay	C_L = 50 pF			110	nsec
	$t_{F(AD)}$	Delay to float				90	nsec
	t_{acm}	Address stable prior to $\overline{MREQ}$ (Memory cycle)		[1]			nsec
	t_{aci}	Address stable prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O cycle)		[2]			nsec
	t_{ca}	Address stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$		[3]			nsec
	t_{caf}	Address stable from $\overline{RD}$ or $\overline{WR}$ during float		[4]			nsec
D_{0-7}	$t_{D(D)}$	Data output delay	C_L = 50 pF			150	nsec
	$t_{F(D)}$	Delay to float during write cycle				90	nsec
	$t_{S\Phi(D)}$	Data setup time to rising edge of clock during M1 cycle		35			nsec
	$t_{S\Phi(D)}$	Data setup time to falling edge of clock during M2 to M5		50			nsec
	t_{dcm}	Data stable prior to $\overline{WR}$ (Memory cycle)		[5]			nsec
	t_{dci}	Data stable prior to $\overline{WR}$ (I/O cycle)		[6]			nsec
	t_{cdf}	Data stable from $\overline{WR}$		[7]			
	t_H	Any hold time for setup time				0	nsec
MREQ	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay from falling edge of clock, $\overline{MREQ}$ low	C_L = 50 pF			85	nsec
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay from rising edge of clock, $\overline{MREQ}$ high				85	nsec
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay from falling edge of clock, $\overline{MREQ}$ high				85	nsec
	$t_{w(\overline{MRL})}$	Pulse width, $\overline{MREQ}$ low		[8]			nsec
	$t_{w(\overline{MRH})}$	Pulse width, $\overline{MREQ}$ high		[9]			nsec
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay from rising edge of clock, $\overline{IORQ}$ low	C_L = 50 pF			75	nsec
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay from falling edge of clock, $\overline{IORQ}$ low				85	nsec
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay from rising edge of clock, $\overline{IORQ}$ high				85	nsec
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay from falling edge of clock, $\overline{IORQ}$ high				85	nsec
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay from rising edge of clock, $\overline{RD}$ low	C_L = 50 pF			85	nsec
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay from falling edge of clock, $\overline{RD}$ low				95	nsec
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay from rising edge of clock, $\overline{RD}$ high				85	nsec
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay from falling edge of clock, $\overline{RD}$ high				85	nsec

Tabella 3 (SGS)

CARATTERISTICHE DINAMICHE (seguito)

Segnale	Simbolo	Parametro	Condizioni di misura	Min.	Tip.	Max.	Unità
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay from rising edge of clock, $\overline{WR}$ low	C_L = 50 pF			65	nsec
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay from falling edge of clock, $\overline{WR}$ low				80	nsec
	$t_{DH\overline{\Phi}(WR)}$	$\overline{WR}$ Delay from falling edge of clock, $\overline{WR}$ high				80	nsec
	$t_{w(\overline{WR}L)}$	Pulse width, $\overline{WR}$ low		[10]			nsec
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay from rising edge of clock, $\overline{M1}$ low	C_L = 50 pF			100	nsec
	$t_{DH(M1)}$	$\overline{M1}$ Delay from rising edge of clock, $\overline{M1}$ high				100	nsec
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay from rising edge of clock, $\overline{RFSH}$ low	C_L = 50 pF			130	nsec
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay from rising edge of clock, $\overline{RFSH}$ high				120	nsec
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ setup time to falling edge of clock			70		nsec
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay time from falling edge of clock	C_L = 50 pF			300	nsec
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ setupt time to rising edge of clock		80			nsec
$\overline{NMI}$	$t_{w(\overline{NM}L)}$	Pulse width, $\overline{NMI}$ low		80			nsec
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ setup time to rising edge of clock		50			nsec
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay from rising edge clock, $\overline{BUSAK}$ low	C_L = 50 pF			100	nsec
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay from falling edge of clock, $\overline{BUSAK}$ high				100	nsec
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ setup time to rising edge of clock		60			nsec
	$t_{F(C)}$	Delay to float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)				80	nsec
	t_{mr}	M1 stable prior to IORQ (Interrupt Ack.)		[11]			nsec

NOTE:

A. Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when $\overline{M1}$ and $\overline{IORQ}$ are both active.
B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.
C. The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.
D. Output delay vs. loaded capacitance: T_{amb} = 70°C V_{CC} = +5V ± 5%.
Add 10 nsec delay for each 50 pF increase in load up to a maximum of 200 pF for the data bus & 100 pF for address & control lines.
E. Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 μsec maximum.

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$
[2] $t_{aci} = t_c - 80$
[3] $t_{ca} = t_{w(\Phi L)} + t_r - 40$
[4] $t_{caf} = t_{w(\Phi L)} + t_r - 60$
[5] $t_{dcm} = t_c - 210$
[6] $t_{dci} = t_{w(\Phi L)} + t_r - 210$
[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 80$
[8] $t_{w(MRL)} = t_c - 40$
[9] $t_{w(MRH)} = t_{w(\Phi H)} + t_f - 30$
[10] $t_{w(\overline{WR}L)} = t_c - 40$
[11] $t_{mr} = 2\,t_c + t_{w(\Phi H)} + t_f - 80$
[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

Tabella 4 (SGS)

DIAGRAMMA DELLE TEMPORIZZAZIONI

I tempi indicati sono stati misurati alle seguenti tensioni:
(se non diversamente specificato)

	"1"	"0"
CLOCK	V_{CC}-0.6V	0.45V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV	±0.5V

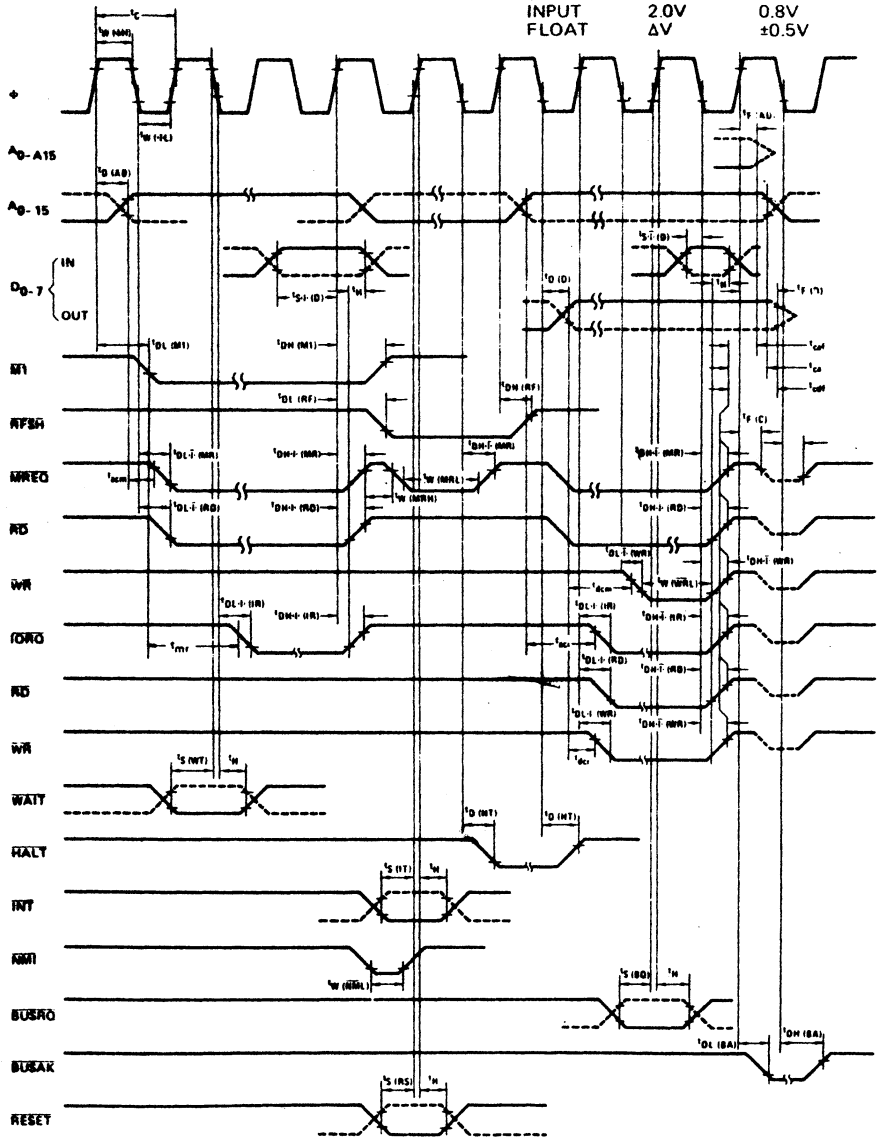


Tabella 5 (SGS)

7. Tecnologie a confronto

Nella tabella 6 sono messe a confronto alcune fra le principali tecnologie utilizzate nella realizzazione dei circuiti integrati per microprocessori.

TECNOLOGIA	DENSITÀ (porte /mm^2)	PRODOTTO veloc. × dissip. (mW·ns)	TEMPO per μ istruz.
P-M0S	90	145	100 μs
N-M0S	150	35	0,5 ÷ 10 μs
C-M0S	50	0,5	0,5 ÷ 10 μs
TTL Schottky	30	30	50 ÷ 100 ns
I^2L	150	150	10 ÷ 50 ns

Tabella 6

Nella tabella 7, sono invece mostrate le tecnologie realizzative di alcuni tra i più noti μP esistenti sul mercato.

TECNOLOGIA	BIT-SLICE	4 BIT	8 BIT	16 BIT
P-M0S	PPS 4 TMS 1000	4040	PPS 8	IMP 16
N-M0S			6800 8080 Z 80 26500 8048	68000 Z 8000 9900 CP 1600
C-M0S	TMS 1000		COSMAC 8085 A 8048	
TTL SCHOTTKY	74S 841 AM 2900 3002			
I_2L		SBP 0400		9445 9900A

Tabella 7

Come si osserva, la tecnologia più utilizzata è la N-M0S che consente di ottenere un buon compromesso tra velocità, potenza dissipata e densità di impacchettamento.

In forte sviluppo risulta anche la tecnologia C-M0S che, se la velocità non è di primaria importanza, offre soprattutto i vantaggi di una bassa dissipazione di potenza a frequenze non troppo elevate, tensione di alimentazione variabile, elevata immunità al rumore ed elevata affidabilità.

Quando invece, la velocità è il fattore predominante (clock maggiore di 10 MHz), si ricorre a dispositivi in tecnologia bipolare tipo I^2L, che unisce ad una elevata velocità di esecuzione (poche decine di ns per porta), una bassa dissipazione di potenza (qualche nw per porta) ed un'elevata densità.

ESERCIZI PROPOSTI

1. Nell'esempio n° 1 di trasferimento punto a punto, si determini la durata dell'impulso Q_A in uscita dal contatore e la massima durata ammissibile per l'impulso STB.
2. Si disegni un diagramma delle temporizzazioni per la lettura, che sia compatibile con la memoria 2112-A dell'esempio n°2.
3. Un circuito in tecnologia TTL è composto da un buffer tri-state che comanda un inverter che a sua volta pilota un diodo led, secondo lo schema di figura 30.

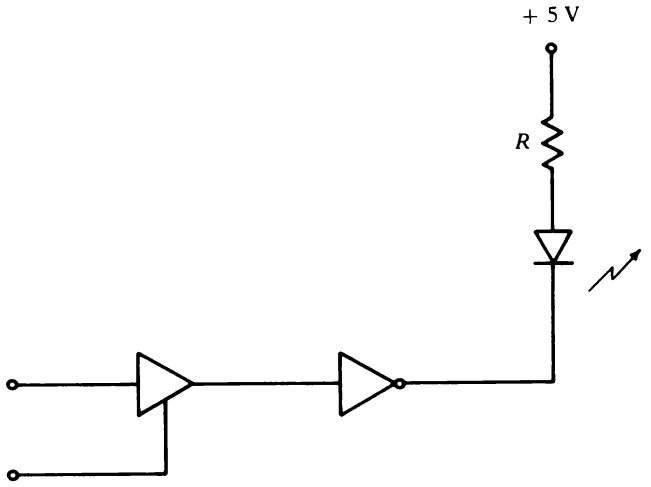


FIG. 30. - Diodo Led pilotato da un inverter.

Considerando che l'ingresso di abilitazione del tri-state è tale da portarlo nello stato di alta impedenza, determinare se il diodo sarà acceso o spento.

4. Un sistema, costituito dal registro IR a due bit, e dai registri A e B ad 8 bit, deve eseguire le seguenti istruzioni:

per IR = 00 ALT
per IR = 10 A - B → A
per IR = 11 A + B → A ma se c'è overflow esegue ALT.

Determinare le microistruzioni e le sequenze dei segnali di abilitazione.

5. Si scriva la tabella degli stati per il progetto della rete sequenziale che esegue il diagramma di flusso di figura 15.

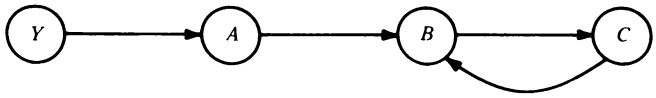


FIG. 31. - Diagramma di flusso di figura 15.

6 .Si disegni lo schema a blocchi di un sistema, completo dei segnali di abilitazione, che realizzi la divisione per un numero qualsiasi n ad otto bit.

7. Si specifichi il contenuto delle fasi di fetch e di execute durante lo svolgimento del seguente programma:

indirizzo	cod. op.		
0100	21	00	02
0103	11	01	02
0106	01	64	00
0109	ED	B0	
010B	76		

8. Facendo uso delle caratteristiche dinamiche relative allo Z 80 riportate nel testo, si disegni il diagramma delle temporizzazioni, relative alla frequenza di 1 MHz, per il ciclo di lettura di codice operativo per l'istruzione NEG.

9. Come l'esercizio 8 ma per un ciclo di scrittura in memoria per l'istruzione LD r, (HL).

10. Un alimentatore ha una caratteristica della tensione di uscita che passa da zero al valore nominale secondo l'andamento di figura 32.

Si calcoli il valore della costante di tempo del circuito R-C, utilizzato

per il reset automatico, se l'ingresso deve rimanere al di sotto della tensione di soglia per almeno altri 10 ms.

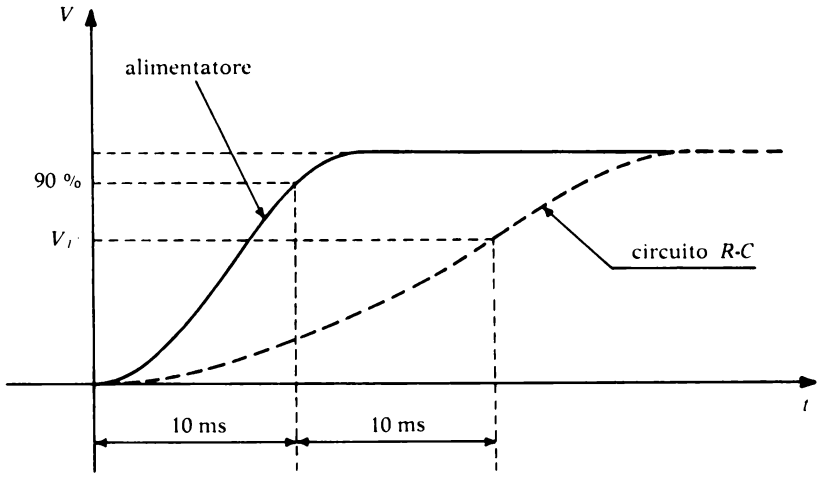


FIG. 32. - Andamento della tensione di reset desiderata.

CAPITOLO QUARTO

DISPOSITIVI DI INPUT-OUTPUT E CIRCUITI D'INTERFACCIA

1. Dispositivi d'Ingresso-Uscita (I/O)

Oltre alla unità centrale di elaborazione e alle varie memorie, un sistema basato su microprocessore deve poter colloquiare con un elemento esterno chiamato normalmente *periferico*.

Il trasferimento dei dati tra un microcomputer ed un periferico è detto Input se l'informazione è diretta dal periferico verso la CPU, oppure Output in caso contrario.

Esempi di unità periferiche sono le tastiere, i terminali video, le teletype, le stampanti, gli switches etc.

Collegamento dei dispositivi di Input/Output (I/O)

Tutte le volte che un dispositivo periferico deve essere collegato al bus dati di un sistema a microprocessore occorre servirsi di un porto di I/0, di solito un buffer tri-state monodirezionale o bidirezionale a seconda del ti-

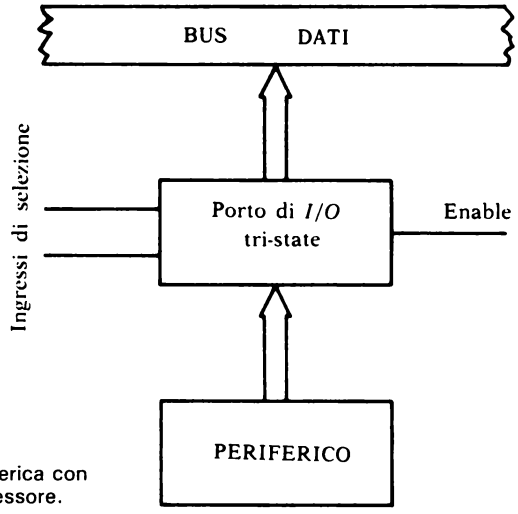


FIG. 1. - Collegamento di una unità periferica con il bus dei dati di un sistema a microprocessore.

po di periferico utilizzato. Ciò si rende necessario perché il dispositivo periferico non può essere collegato permanentemente al bus dati, in quanto su di esso viaggiano informazioni inerenti la memoria, ma deve essere collegato soltanto quando è stato selezionato e abilitato a comunicare con il sistema. In fig. 1 è mostrato il collegamento tra una unità periferica ed un sistema a microcomputer utilizzando un porto di I/0.

Di solito i dispositivi periferici vengono selezionati mediante gli otto bit meno significativi del bus degli indirizzi (da A_0 ad A_7) che permettono la selezione di $2^8 = 256$ diversi porti di I/0. Se il numero dei periferici dovesse essere maggiore di 256 è sempre possibile considerarli come celle di memoria. Ciascun periferico inoltre può essere visto dalla CPU come un registro su cui si può scrivere o leggere un dato. Chiaramente le operazioni di I/0 per poter essere eseguite debbono essere sincronizzate con le richieste o l'invio dei dati da parte del dispositivo periferico.

Così, per esempio, se un sistema a microprocessore deve leggere un dato da una tastiera deve poter sapere quando il tasto è stato premuto. La sincronizzazione può essere effettuata abbastanza semplicemente mediante l'introduzione di un bit di stato che è settato dal segnale di dato pronto proveniente dall'esterno e resettato durante la lettura fatta dalla CPU. Le operazioni di I/0 possono quindi essere pensate come operazioni di lettura/scrittura in memoria in cui l'indirizzo è quello del periferico selezionato.

Tecnica di collegamento Memory-Mapped I/0.

La tecnica di gestione degli I/0 mediante il metodo Memory-Mapped I/0 non richiede istruzioni di input-output dedicate nè particolari linee di controllo in quanto il periferico selezionato è considerato come una normale locazione di memoria. Si possono così usare per le operazioni di I/0 tutte le normali istruzioni del microprocessore che interessano la memoria. L'inconveniente più grande di questo modo di operare è dato dall'utilizzo di una zona di I/0 normalmente riservata all'indirizzamento della memoria del sistema (fig. 2).

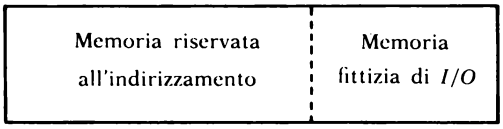


FIG. 2. -

In fig. 3 è mostrato lo schema di un dispositivo che permette la scrittura di un dato proveniente dall'esterno in un registro della CPU utilizzando la tecnica di gestione di I/0 Memory Mapped.

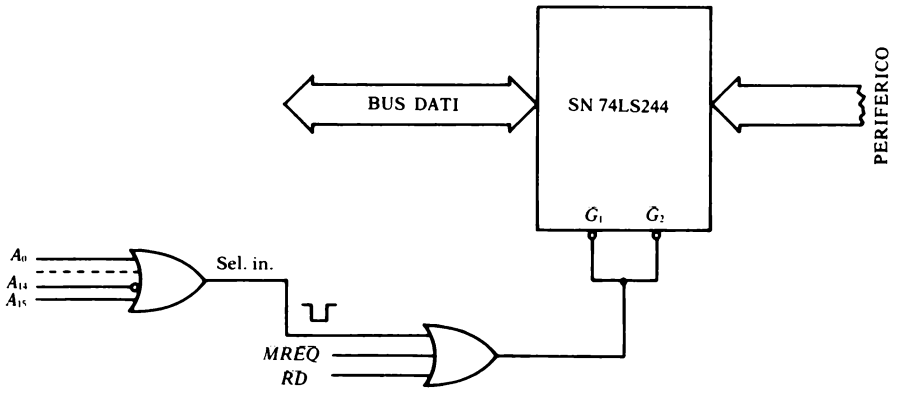


FIG. 3. - Lettura di un dato da un periferico mediante la tecnica Memory Mapped I/O.

Il funzionamento del circuito è il seguente:
supposto che la memoria del sistema sia di 8 K (da 0000_H a 2000_H) al periferico può essere assegnato uno qualsiasi degli indirizzi a partire da 2000_H, ad esempio 4000_H.

Non appena viene eseguita l'istruzione LDA, (4000) l'uscita della porta logica 0R a tre ingressi diventando bassa abilita il porto d'ingresso e permette così il trasferimento dell'informazione dal periferico al registro A della CPU (fig. 5).

La selezione dell'indirizzo è stata effettuata negando la linea del bus degli indirizzi A_{14} in modo da attivare il buffer tri-state soltanto per la configurazione di indirizzo 0100 0000 0000 0000 = 4000_H, cioè per l'indirizzo assegnato al dispositivo periferico.

La decodifica di tutti i 16 bit del bus degli indirizzi ha il vantaggio di occupare un solo byte di memoria fittizia per il porto di I/0 (64K: $2^{16} = 1$ byte) ma comporta un circuito decodificatore piuttosto complesso schematizzato nella figura con un 0R a 16 ingressi. D'altra parte se ad esempio

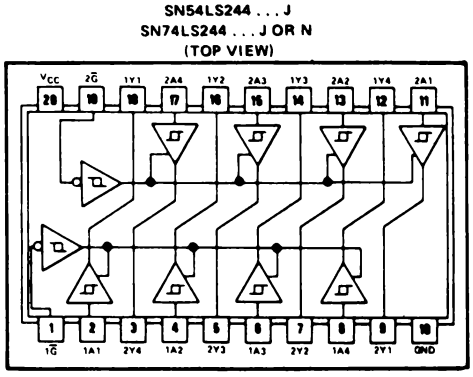


FIG. 4 - Piedinatura e caratteristiche dell'SN 74LS244 (Texas).

recommended operating conditions

PARAMETER	SN54LS'			SN74LS'			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC} (see Note 1)	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−12			−15	mA
Low-level output current, I_{OL}			12			24	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

NOTE 1: Voltage values are with respect to network ground terminal.

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS†		SN54LS'			SN74LS'			UNIT
				MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage			2			2			V
V_{IL}	Low-level input voltage					0.7			0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN,	I_I = −18 mA			−1.5			−1.5	V
	Hysteresis ($V_{T+} - V_{T-}$)	V_{CC} = MIN		0.2	0.4		0.2	0.4		V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IL} = 0.8 V,	V_{IH} = 2 V, I_{OH} = MAX	2.4	3.4		2.4	3.4		V
		V_{CC} = MIN, V_{IL} = 0.5 V,	V_{IH} = 2 V, I_{OH} = MAX	2			2			
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL}max	I_{OL} = 12 mA			0.4			0.4	V
			I_{OL} = 24 mA						0.5	
I_{OZH}	Off-state output current, high-level voltage applied	V_{CC} = MAX, V_{IH} = 2 V, V_{IL} = V_{IL}max	V_O = 2.7 V			20			20	μA
I_{OZL}	Off-state output current, low-level voltage applied		V_O = 0.4 V			−20			−20	
I_I	Input current at maximum input voltage	V_{CC} = MAX,	V_I = 7 V			0.1			0.1	mA
I_{IH}	High-level input current, any input	V_{CC} = MAX,	V_I = 2.7 V			20			20	μA
I_{IL}	Low-level input current	V_{CC} = MAX,	V_{IL} = 0.4 V			−0.2			−0.2	mA
I_{OS}	Short-circuit output current§	V_{CC} = MAX		−50		−225	−50		−225	mA
I_{CC}	Supply current	Outputs high	V_{CC} = MAX, Outputs open	All		13	23	13	23	mA
		Outputs low		'LS240		26	44	26	44	
				'LS241, 'LS244		27	46	27	46	
		All outputs disabled		'LS240		29	50	29	50	
				'LS241, 'LS244		32	54	32	54	

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.

‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.

§ Not more than one output should be shorted at a time, and duration of the short-circuit should not exceed one second.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER		TEST CONDITIONS		'LS240			'LS241, 'LS244			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low-to-high-level output	C_L = 45 pF, See Note 2	R_L = 667 Ω,		9	14		12	18	ns
t_{PHL}	Propagation delay time, high-to-low-level output				12	18		12	18	ns
t_{PZL}	Output enable time to low level				20	30		20	30	ns
t_{PZH}	Output enable time to high level				15	23		15	23	ns
t_{PLZ}	Output disable time from low level	C_L = 5 pF, See Note 2	R_L = 667 Ω,		15	25		15	25	ns
t_{PHZ}	Output disable time from high level				10	18		10	18	ns

NOTE 2: Load circuit and voltage waveforms are shown on page 3-11.

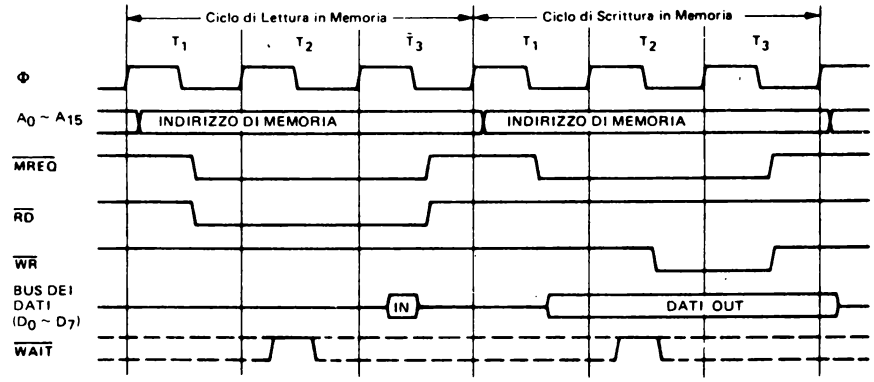


FIG. 5. - Temporizzazioni durante l'esecuzione di una istruzione di caricamento (dal manuale tecnico SGS).

la decodifica di indirizzo fosse stata effettuata utilizzando soltanto le linee A_{15} ed A_{14} mediante una porta logica Nand, il porto di I/0 occuperebbe una memoria fittizia pari a 64K: 2^2 = 16K, cioè ogni indirizzo compreso fra COOO e FFFF selezionerebbe il periferico (fig. 6).

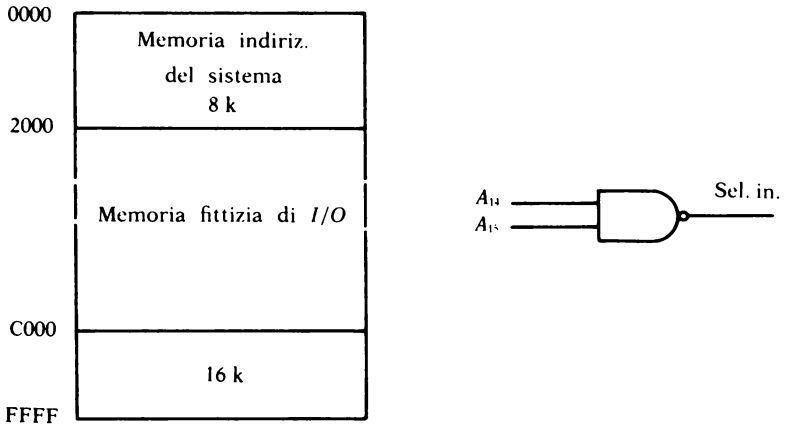


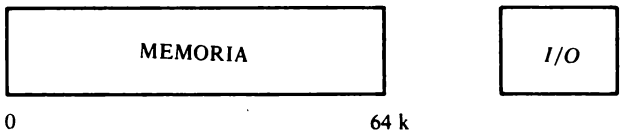
FIG. 6.

Tecnica di collegamento I/0 isolato (I/O Mapped I/O)

Nella tecnica di collegamento di I/0 isolato le operazioni d'ingresso e di uscita sono realizzate mediante istruzioni dedicate di input e di output. Queste istruzioni permettono di rendere fisicamente distinte le zone di I/0 dalle zone di memoria (fig. 7) mediante segnali di controllo che si attivano

durante l'esecuzione delle stesse istruzioni. Con questo tipo di tecnica è possibile selezionare non più di 256 porti di I/0, ciò comporta una evidente limitazione rispetto alla tecnica di I/0 memory-mapped.

FIG. 7.



Nella fig. 8 è mostrato un dispositivo capace di selezionare quattro porti di I/0 e di leggere i dati presentati dai relativi periferici mediante la tecnica di I/0 isolato.

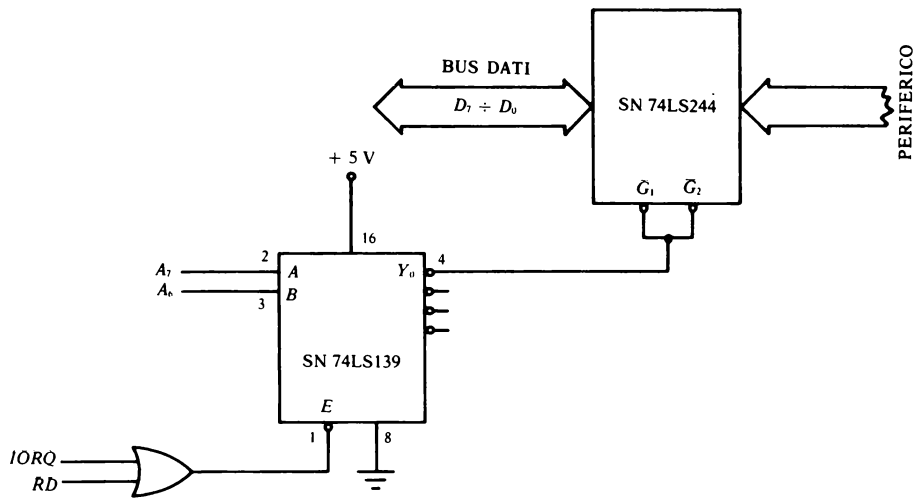


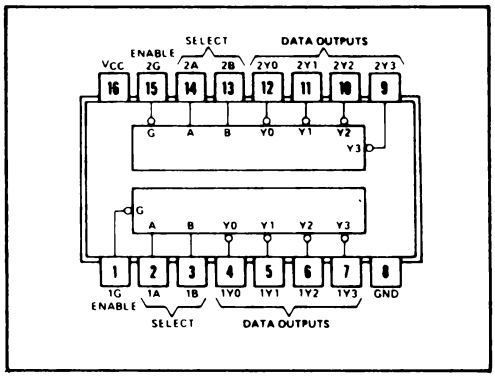
FIG. 8. - Lettura dei dati inviati da quattro periferici con la tecnica dell'I/O isolato.

Il funzionamento del circuito è il seguente: non appena viene eseguita l'istruzione IN A, (n) i pin $\overline{IORQ}$ e $\overline{RD}$ della CPU si portano al livello logico basso (fig. 10) ed il demultiplexer SN 74LS139 (Texas) viene abilitato.

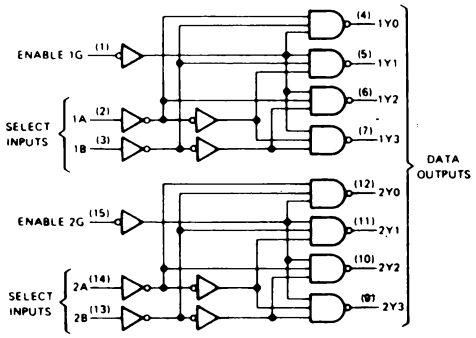
A seconda dello stato di A_7 e A_6 viene selezionata una delle quattro uscite Y_0, Y_1, Y_2 e Y_3. Così ad esempio se si volesse selezionare il dispositivo periferico collegato con l'uscita Y_0 basterà, tenendo conto della tabella della verità del demultiplexer, che i due bit più significativi di n siano uguali a zero (con n = 00XX XXXX ed X 0 od 1 indifferentemente).

La selezione di una delle uscite del demultiplexer abilita il buffer tristate ad essa collegato a trasferire sul bus dei dati l'informazione inviata

SN54LS139, SN54S139 . . . J OR W PACKAGE
SN74LS139, SN74S139 . . . J OR N PACKAGE
(TOP VIEW)



'LS139, 'S139



'LS139, 'S139
(EACH DECODER/DEMULTIPLEXER)
FUNCTION TABLE

INPUTS			OUTPUTS			
ENABLE	SELECT					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = high level, L = low level, X = irrelevant

FIG. 9. - Piedinatura, schema funzionale e tabella della verità del demultiplexer SN 74LS139 (Texas).

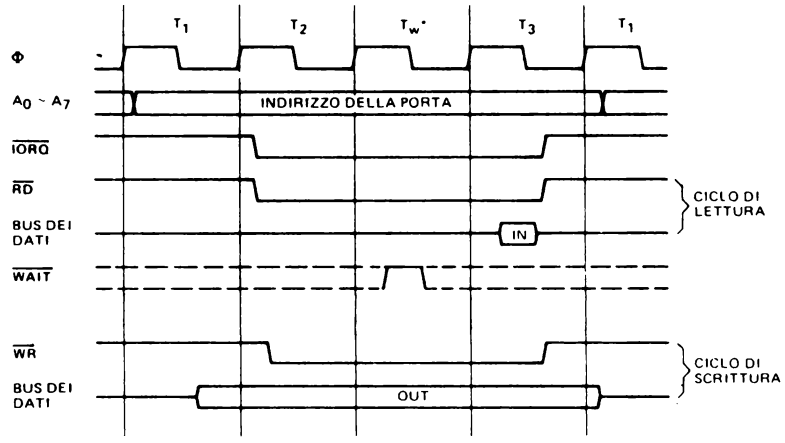


FIG. 10. - Temporizzazioni durante l'esecuzione di una istruzione d'ingresso (dal manuale tecnico SGS).

dal periferico corrispondente, informazione che viene caricata automaticamente nell'accumulatore della CPU.

In definitiva ecco cosa accade per esempio quando viene eseguita l'istruzione IN A, (40):

— $\overline{IORQ}$ e $\overline{RD}$ sono attivati
— sulla parte bassa del bus degli indirizzi è posto 0100 0000 (40_H).
— A = 0 B = 1, è selezionata l'uscita Y_2 ed abilitato il buffer tri-state ad essa collegato.
— l'accumulatore pone il suo contenuto, nella parte alta del bus degli indirizzi (da A_7 ad A_{15}) e immagazzina il byte di dati del periferico selezionato.

2. L'interrupt (Interruzione)

L'interrupt è un intervento esterno sul programma principale provocato da un periferico. Una volta effettuata la richiesta d'interruzione l'esecuzione del programma principale viene interrotta e la CPU esegue la subroutine o routine di servizio dell'interrupt per poi ritornare al programma principale nel punto in cui era stato sospeso una volta completato il programma.

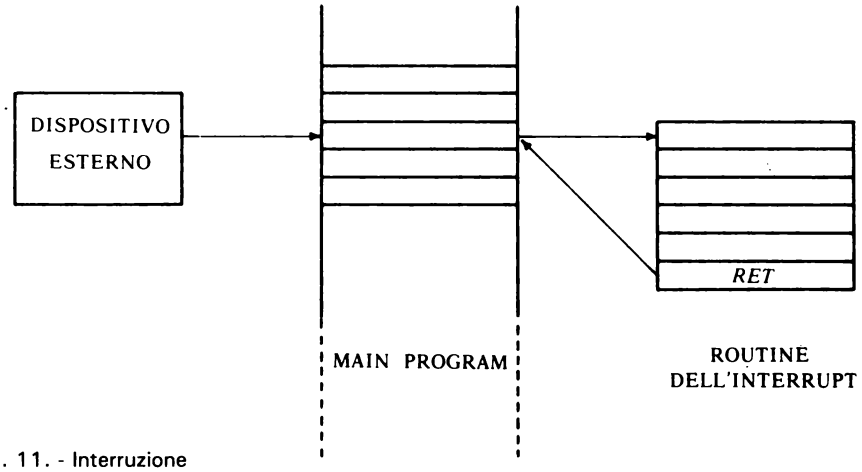


FIG. 11. - Interruzione

L'interruzione del programma principale può avvenire o da parte di un solo periferico oppure, più frequentemente, da più periferici collegati al bus dei dati della CPU (fig. 12).

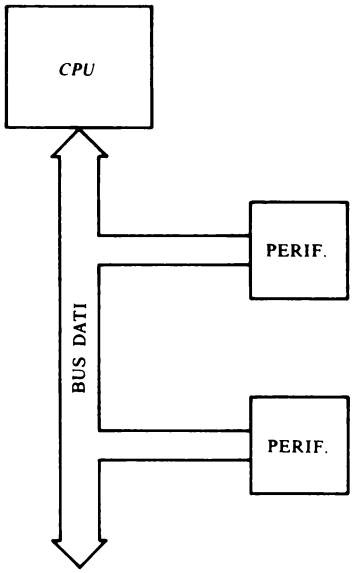


FIG. 12.

In quest'ultimo caso l'interruzione può avvenire in due diversi modi come mostrato nella fig. 13.

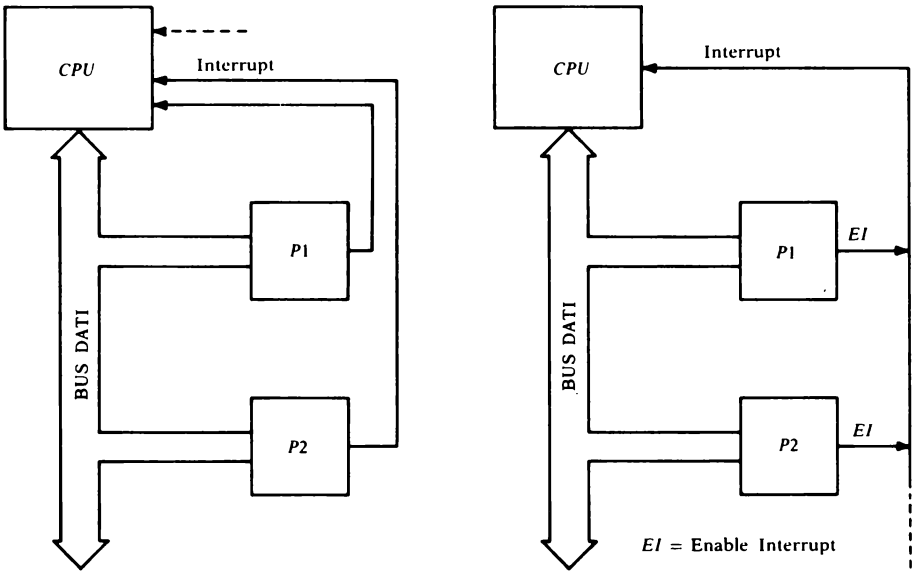


FIG. 13. - Controlli di priorità dell'interruzione: (a) centralizzata, (b) distribuita.

Nel caso della fig. 13a si parla di *gestione centralizzata dell'interruzione.* In questo caso l'unità centrale di elaborazione viene collegata al dispositivo esterno mediante una propria linea di interrupt, ne segue che sono necessari da parte della CPU tanti pin d'interrupt quanti sono i dispositivi esterni ad essa collegati.

La figura 13b mostra invece una *gestione distribuita della interruzione* da parte della CPU, così definita in quanto la linea di interruzione è unica per tutti i dispositivi periferici.

La gestione centralizzata dell'interrupt è una tecnica usata di solito nei piccoli sistemi dove il numero dei dispositivi esterni e quindi delle linee d'interrupt risulta contenuto. Nel caso di gestione centralizzata dell'interrupt occorre far capire alla CPU quali dispositivi esterni hanno chiesto l'interruzione, e nel caso di richieste simultanee quale deve essere la priorità d'interrupt da assegnare.

Richiesta d'interruzione

Esistono diversi modi per stabilire quale periferico ha chiesto l'interruzione. Il primo è detto metodo di *polling,* il secondo invece è detto *metodo di vettorizzazione dell'interrupt.* Il metodo di polling consiste in una sequenza di test sullo stato dei dispositivi esterni in modo da individuare chi ha richiesto l'interruzione, in pratica la CPU interroga ciclicamente ogni periferico per determinare se vi è stata una richiesta di interruzione. Tale metodo non è molto usato in quanto distoglie la CPU dal suo normale lavoro anche se in realtà non esistono richieste d'interrupt.

Una volta attivata la linea d'interruzione la CPU va ad interrogare un bit del registro di stato associato ad ogni periferico e se lo trova settato riconosce la richiesta d'interruzione.

Nel metodo di vettorizzazione dell'interrupt è invece il dispositivo esterno che segnala alla CPU la richiesta d'interruzione mediante l'invio di un byte nel bus dei dati cosicchè la CPU gestisce l'interrupt soltanto se è stato richiesto mediante l'attivazione della linea di riconoscimento dell'interruzione (interrupt acknowledge). Chiaramente ogni dispositivo esterno deve essere caratterizzato da un byte personalizzato detto *interrupt vector.* Una volta riconosciuta l'interruzione il microprocessore esegue la routine d'interrupt per poi ritornare al programma principale.

Priorità

Anche per stabilire la priorità, nel caso vi siano più richieste simultanee d'interruzione da parte di dispositivi periferici, esistono due diversi criteri:

1) Controllo centralizzato di priorità
2) Controllo distribuito di priorità o daisy chain (abilitazione a catena).

Nel controllo centralizzato la priorità di interruzione consiste nell'assegnare ad ogni dispositivo periferico una priorità crescente. Ciò può essere fatto abbastanza semplicemente mediante un codificatore di priorità, qua-

le ad esempio l'SN 74 LS 147 (Texas), collegando l'ingresso a priorità più alta del codificatore con il periferico più prioritario e l'ingresso a priorità più bassa con quello meno prioritario (fig. 14).

'147, 'LS147
FUNCTION TABLE

INPUTS									OUTPUTS			
1	2	3	4	5	6	7	8	9	D	C	B	A
H	H	H	H	H	H	H	H	H	H	H	H	H
X	X	X	X	X	X	X	X	L	L	H	H	L
X	X	X	X	X	X	X	L	H	L	H	H	H
X	X	X	X	X	X	L	H	H	H	L	L	L
X	X	X	X	X	L	H	H	H	H	L	L	H
X	X	X	X	L	H	H	H	H	H	L	H	L
X	X	X	L	H	H	H	H	H	H	L	H	H
X	X	L	H	H	H	H	H	H	H	H	L	L
X	L	H	H	H	H	H	H	H	H	H	L	H
L	H	H	H	H	H	H	H	H	H	H	H	L

H = high logic level, L = low logic level, X = irrelevant

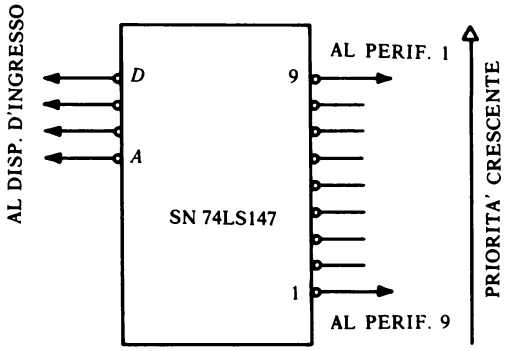


FIG. 14. - Controllo centralizzato di priorità mediante l'integrato SN 74LS147 (Texas).

In seguito il codificatore di priorità provvede a trasmettere il codice binario corrispondente al periferico più prioritario, fra quelli che hanno richiesto l'interruzione, ad un opportuno dispositivo d'ingresso che dà il via alla routine di gestione dell'interrupt mediante la quale viene ricavato l'indirizzo di partenza della subroutine d'interruzione relativa al periferico selezionato.

Nei sistemi che prevedono l'interrupt vettorizzato viene usato di solito il controllo distribuito di priorità. In tale tipo di controllo la richiesta d'interruzione da parte di un periferico può essere soddisfatta soltanto se non sono in corso richieste di interruzione formulate da elementi a priorità più alta.

Riferendosi alla fig. 15 se la periferica P_1 (a priorità più elevata) richiede l'interruzione tutte le altre vengono disabilitate cosicché eventuali altre richieste di interruzione possono essere soddisfatte soltanto alla fine della esecuzione della subroutine d'interrupt.

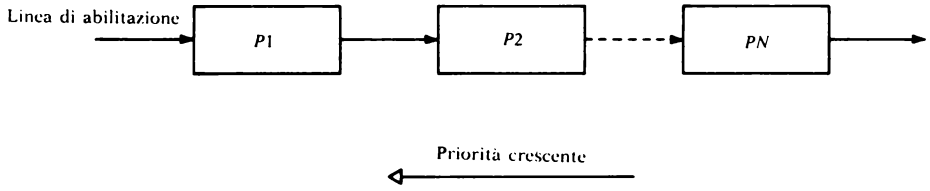


FIG. 15. - Controllo di priorità «daisy chain».

Se ad esempio la periferica P_2 chiede l'interruzione, quando P_1 è disabilitata, la linea di abilitazione dell'interruzione rimane attiva fino a P_2 escludendo tutte le altre periferiche alla sua destra anche se da parte di quest'ultime c'è stata una richiesta di interruzione. La gerarchia di priorità in questo caso è data soltanto dalla mutua posizione dei vari periferici cosicché tale struttura si presenta molto rigida in quanto una variazione di priorità comporta un diverso allineamento fisico dei periferici interessati.

Nel controllo distribuito di priorità inoltre viene introdotto un certo tipo di ritardo di propagazione, variabile a seconda del numero dei periferici utilizzati, dovuto alle porte logiche collegate in cascata che costituiscono la linea di abilitazione. Di conseguenza occorre ritardare la lettura del vettore d'interruzione della periferica richiedente l'interrupt fino a quando il segnale di abilitazione si è propagato per tutta la linea.

Questo può essere fatto abbastanza semplicemente per mezzo di buffer invertenti e non a seconda dei casi. Un vantaggio del controllo distribuito di priorità rispetto a quello centralizzato è quello di avere una unica linea d'interrupt.

In fig. 16 è riportato lo schema completo di un circuito capace di gestire un dispositivo con interruzione vettorizzata mediante il controllo daisy chain.

Il funzionamento è il seguente:

se ad un certo istante il periferico P_1 richiede l'interruzione mediante l'attivazione della linea di interrupt $\overline{INR1}$ (interrupt request, attiva bassa) l'ingresso 1 della porta logica And si porta il livello logico basso e così pure la sua uscita.

Viene quindi attivata la linea di interruzione $\overline{INT}$. La CPU risponde a questa richiesta d'interruzione mediante un opportuno segnale di riconoscimento dell'interruzione inviato sulla linea $\overline{INTA}$ (interrupt acknowledge: riconoscimento della interruzione) cosicché l'uscita della porta logica Or collegata a P_1 diventa bassa. A questo punto il periferico P_1 risulta abilitato e può inviare sul bus dei dati il vettore di riconoscimento dell'interruzione che lo identifica e permette alla CPU l'esecuzione della routine di servizio dell'interrupt. Se altri periferici oltre a P_1 hanno effettuato simultaneamente richieste d'interruzione, tali richieste non possono essere soddisfatte da parte della CPU. In queste condizioni infatti la richiesta d'interruzione da parte di P_1 porta l'uscita IE01 e quindi l'ingresso IEI2 al livello logico basso cosicché gli ingressi $\overline{EI}$ (input enable) essendo alti tengono disabilitati i rispettivi periferici. Una volta terminata la routine di servizio dell'interrupt relativa a P_1 se P_2 sta ancora chiedendo l'interruzione ($\overline{INR2} = 0$) tale richiesta può essere soddisfatta in quanto in queste condizioni si ha che:

$$\overline{INR1} = 1 \qquad \overline{IEI12} = \overline{EI2} = 0$$

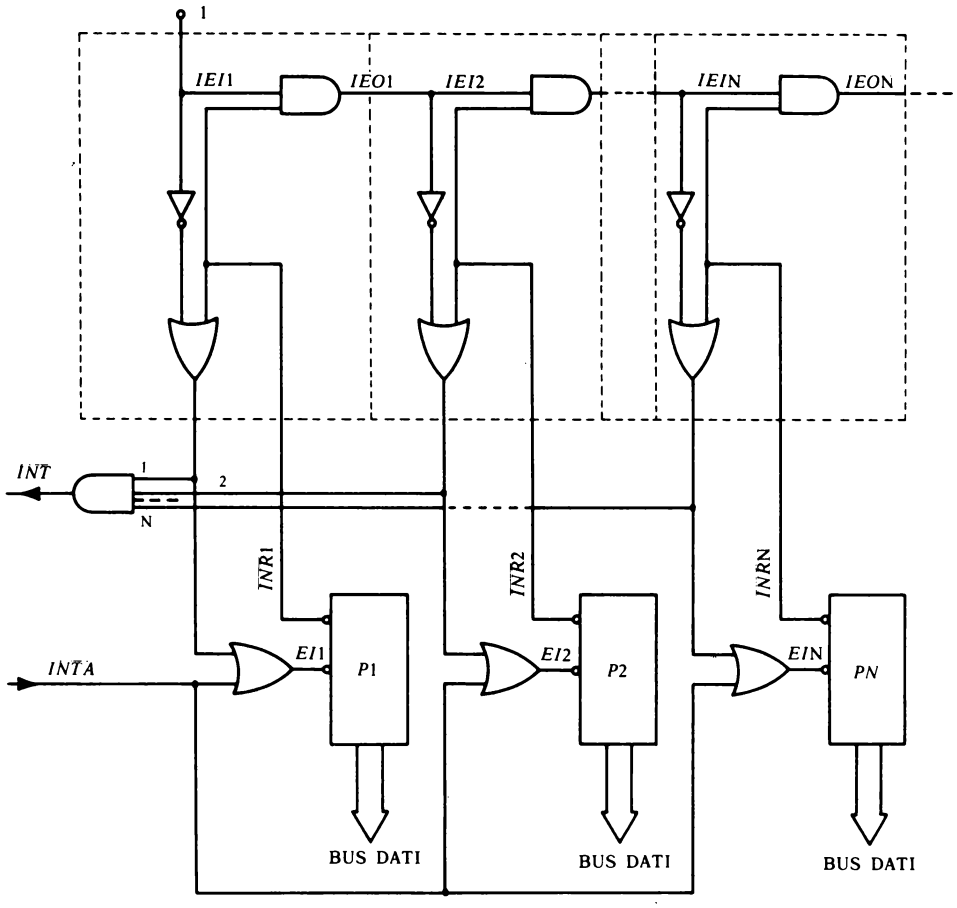


FIG. 16. - Interruzione vettorizzata mediante il controllo daisy chain.

Naturalmente poiché in questo caso $\overline{IE02} = ... \overline{IEIN} = \overline{IE0N} = 0$ tutti gli altri periferici risultano disabilitati, anche se hanno richiesto l'interruzione tramite l'attivazione delle linee $\overline{INR}$, in quanto gli ingressi di abilitazione $\overline{EI}$ sono al livello logico alto.

Interruzioni mascherabili e non mascherabili

Le interruzioni in un sistema a microprocessore possono essere suddivise in interruzioni mascherabili ed in interruzioni non mascherabili. Per interruzione *mascherabile* si intende una interruzione che può essere ignorata dalla CPU se quest'ultima non è stata abilitata ad eseguirla mediante opportune istruzioni. Una interruzione è detta *non mascherabile* quando viene eseguita indipendentemente dal software del sistema.

3. Le interruzioni nel microprocessore Z 80

Nei sistemi utilizzanti come CPU lo Z 80 le richieste di interruzione mascherabili, inviate sulla linea di richiesta d'interruzione $\overline{INT}$, possono essere eseguite o non eseguite a seconda che esse siano state abilitate o disabilitate mediante le istruzioni EI (enable interrupt) e DI (disable interrupt) rispettivamente. Lo Z 80 oltre alla linea $\overline{INT}$ è provvisto di un ingresso $\overline{NMI}$ (interruzione non mascherabile) rivolto a richieste di interruzioni non mascherabili. Normalmente tale ingresso risulta disabilitato ($\overline{NMI}$ = 1) in quanto una richiesta di interruzione non mascherabile, prioritaria rispetto a quella mascherabile, avviene soltanto in certe particolari situazioni come ad esempio la caduta di tensione di alimentazione in cui occorre poter salvare lo stato della CPU in modo che al ritorno della tensione la CPU possa riprendere a funzione correttamente.

Interruzioni mascherabili

Se la CPU riceve una richiesta d'interruzione, una volta che sia stata predisposta ad attuarla mediante l'istruzione EI, appena ha completato l'istruzione in corso invia un segnale di abilitazione ($\overline{IORQ}$ durante $\overline{M1}$, fig. 17) se non vi è in corso una richiesta di controllo dei vari bus del microprocessore da parte di dispositivi esterni.

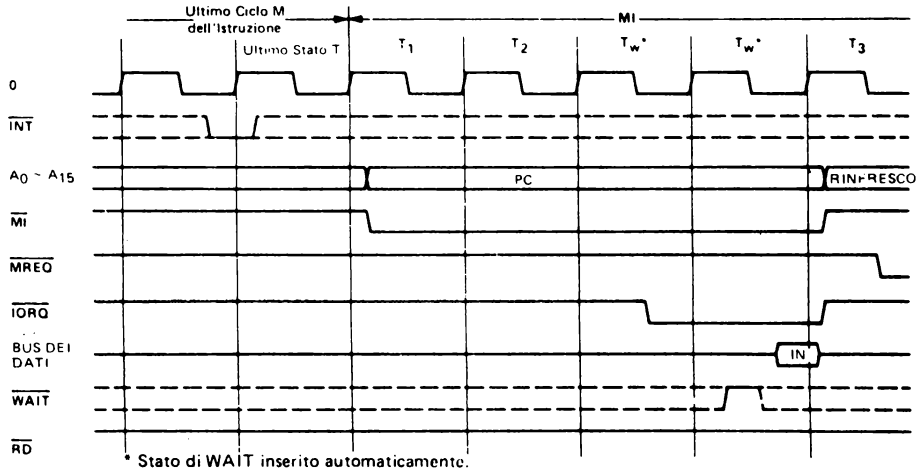


FIG. 17. - Ciclo di richiesta di riconoscimento di interruzione mascherabile (dal manuale tecnico SGS).

La risposta della logica esterna o periferico a questo segnale di riconoscimento dell'interruzione dipende dal modo in cui la CPU è stata predisposta ad operare. I possibili modi di interruzione per lo Z 80 sono tre: modo zero, modo uno e modo due.

Modo Zero

La CPU opera nel modo 0 quando il periferico che causa l'interruzione invia sul bus dati un vettore d'interrupt ad otto bit che è interpretato come istruzione che deve essere eseguita appena terminata quella in corso.

Di solito è una delle istruzioni di restart tra le otto possibili. *Quando la CPU è azzerata essa ($\overline{RESET}$ = 0) si predispone automaticamente nel modo zero di interruzione.* L'istruzione di restart, equivalente al singolo byte di una CALL, salva nello stack il contenuto del program counter (PC) ed esegue la subroutine dell'interruzione caricata all'indirizzo di memoria così calcolato:

Codice operativo
della restart
11 t 111

con
t = 000
t = 001
t = 010
t = 100
t = 101
t = 110
t = 111

Nuovo contenuto del PC
0000 0000 00 t 000

Modo Uno

Nel modo 1 la CPU risponde alla richiesta d'interruzione eseguendo una restart dalla locazione di memoria 0038_H. In questo caso non è necessario nessun vettore d'interruzione da parte del periferico.

Modo Due

Questo è il modo più potente di risposta ad una interruzione da parte dello Z 80. Mediante un singolo byte, generato dal periferico si può generare una chiamata indiretta a qualsiasi locazione di memoria. Quando si opera nel modo 2 il programmatore deve costruirsi una tabella di indirizzi di 16 bit per tutti i programmi di servizio delle interruzioni. Questa tabella può essere collocata in una qualsiasi zona di memoria indirizzabile. Quando la CPU riceve una richiesta di interruzione viene generato un indirizzo a 16 bit che permette di ricavare dalla tabella l'indirizzo della subroutine d'interrupt desiderata. L'indirizzo a 16 bit è formato dal contenuto del registro I, precedentemente caricato, e dal vettore d'interruzione del periferico interessato *il cui bit meno significativo deve essere sempre zero.*

Un esempio di interruzione in modo 2

Si supponga di avere 32 possibili periferici che possono richiedere una interruzione. Ciascun periferico ha una propria subroutine di servizio dell'interruzione, di conseguenza esistono 32 indirizzi di partenza immagazzinati in 64 byte di memoria, uno per ogni periferico. Si assuma per esempio che questi 64 byte siano immagazzinati in una tabella con indirizzi di memoria che vanno da 0800 a 083F, ora in accordo al modo 2 di interruzione occorre caricare nel registro I della CPU il valore esadecimale 08. Successivamente ad una richiesta di interruzione da parte di un periferico segue un segnale di avvenuto riconoscimento dell'interruzione da parte della CPU. A questo punto il periferico interessato (per esempio quello a cui corrisponde il vettore di interruzione 2A) invia il proprio byte di riconoscimento sul bus dei dati (fig. 18).

Da quanto visto finora si può comprendere come il modo di interruzione 1 sia il più rapido fra i tre possibili. Tale modo infatti richiede semplicemente che la prima istruzione del programma di interruzione inizi a partire dalla locazione di memoria 0038_H o che da li tramite una CALL caricata precedentemente venga chiamata la subroutine d'interrupt allocata in una

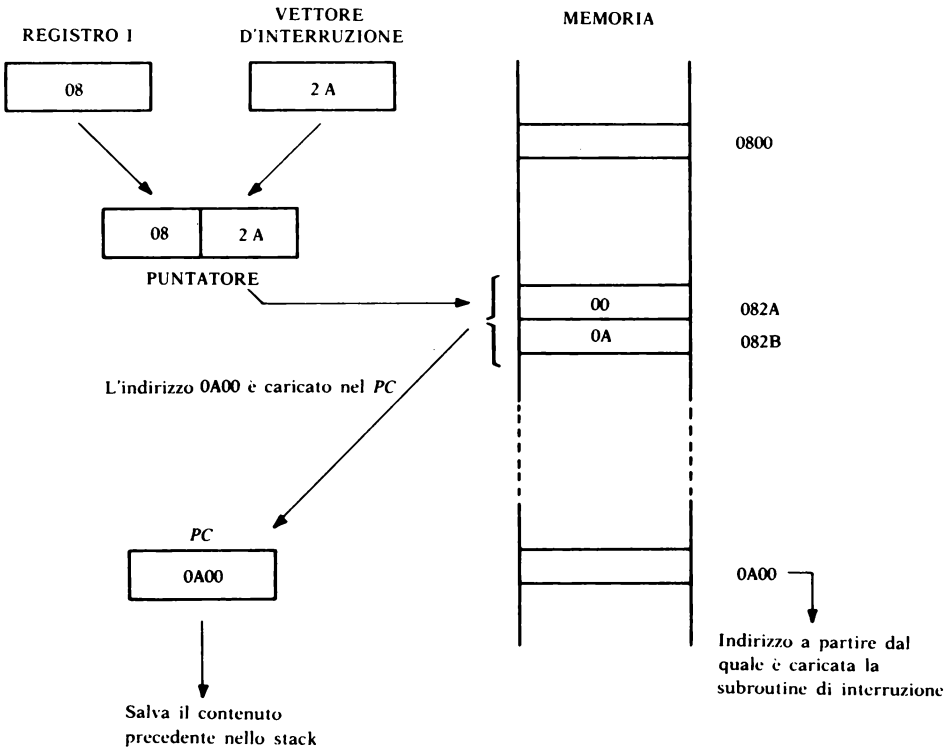


FIG. 18. - Esempio di Interruzione in modo 2.

data zona di memoria. In ogni caso non è mai richiesta una logica esterna per generare un vettore d'interruzione.

Resta però il fatto che operando nel modo 1 è necessaria la generazione di un segnale di richiesta d'interruzione da parte del periferico, e una volta che la richiesta è stata riconosciuta occorre un dispositivo in grado di resettare il periferico richiedente. Tutto questo come si vedrà in seguito può essere fatto mediante un componente LSI quale ad esempio lo Z 80 PIO (Parallel Input/Output Interface).

Interruzione non mascherabile

Il modo di operare dell'interruzione non mascherabile è pressoché simile a quello di modo 1 mascherabile con la sola differenza dell'indirizzo di restart che in questo caso vale 0066 anziché 0038. In fig. 19 è riportata la sequenza di richiesta di una interruzione non mascherabile.

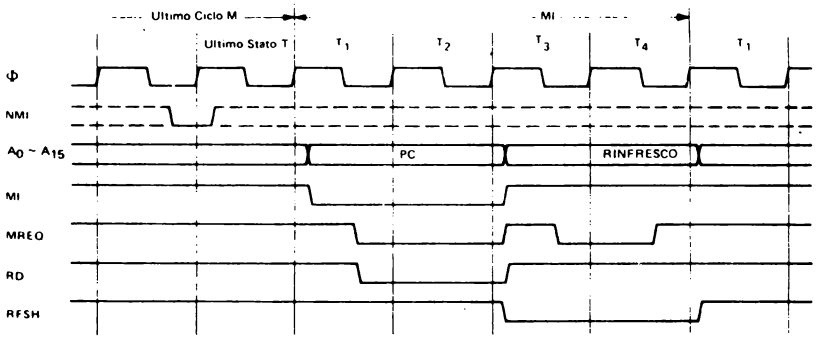


FIG. 19. - Ciclo di riconoscimento di una interruzione non mascherabile (dal manuale tecnico SGS).

4. Esempi software-hardware dei vari modi di interruzione nello Z 80

Modo 0

Si supponga di dover interrompere, non appena un periferico lo richiede, il seguente programma di carattere prettamente didattico e di conseguenza senza particolare significato:

Programma principale

	Codice oggetto	Assembly
P_1 0100	16 00	LD D, 00
	1E FF	LD E, FF
	CB 7A	BIT 7, D
	CB 73	BIT 6, E
	CB D2	SET 2, D
	CB EB	SET 5, E
	1D	DEC E
	C3 00 01	JP P_1

e che inoltre la subroutine d'interrupt del periferico sia:

Subroutine d'interrupt:

06 AA	LD B, AA
0E 88	LD C, 88
A0	AND B
00	NOP
21 77 00	LD HL, 0077

Le istruzioni necessarie da inserire nel programma principale e nella subroutine dell'interrupt sono rispettivamente EI, IM 0 ed EI, RETI.

Istruzione EI (Enable interrupt)

Questa istruzione ha il compito di abilitare la CPU alle interruzioni.

Istruzione DI (Disable interrupt)

L'istruzione DI disabilita le interruzioni. In fig. 20 è mostrato come le istruzioni EI ed DI agiscono all'interno della CPU.

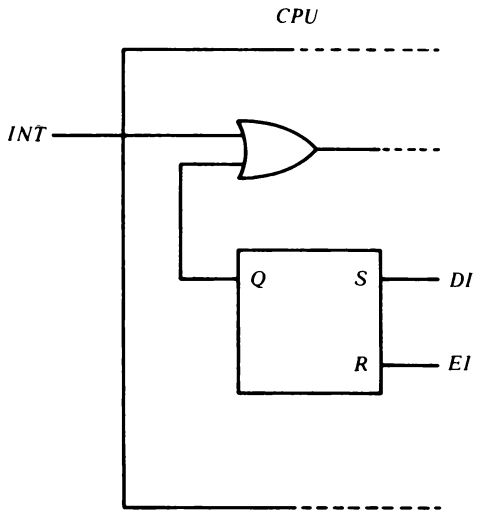


FIG. 20.

Istruzione IM 0 (Interrupt mode 0)

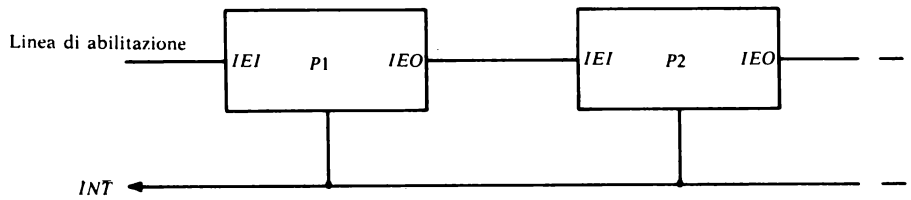
L'istruzione IM 0 fa si che il periferico richiedente l'interruzione possa inserire la sua istruzione, ad uno o a tre byte a seconda che essa sia una restart od una CALL, nel bus dei dati e permettere così l'esecuzione della routine d'interrupt.

Istruzione RETI (Return from interrupt)

L'istruzione RETI viene usata alla fine di una subroutine di servizio dell'interrupt per ripristinare di nuovo il contenuto del program counter e segnalare ai dispositivi periferici che la subroutine di servizio è terminata. Tale istruzione permetterà di lavorare, nel caso dello Z 80 PIO, con una struttura daisy chain (parag. 6).

ESEMPIO:

Sono dati due dispositivi d'interruzione P_1 e P_2 collegati in configurazione daisy-chain, con P_1 prioritario rispetto a P_2.



P_2 genera una interruzione e viene riconosciuto. Di conseguenza l'uscita di abilitazione dell'interrupt IE0 di P_2 si abbassa impedendo a qualsiasi periferico di priorità inferiore di interrompere mentre viene servito P_2. Poi P_1 genera una interruzione, sospendendo il servizio di P_2. A questo punto l'IE0 di P_1 si abbassa indicando che in quel momento viene servito un periferico più prioritario. La routine di P_1 viene completata e una istruzione di RETI resetta l'IE0 di P_1 permettendo alla routine di P_2 di continuare. Una seconda istruzione di RETI completa la routine di P_2 e l'IE0 di P_2 viene riportato alto così da consentire l'accesso all'interruzione da parte di periferici meno prioritari.

È da tener presente che lo Z 80 non accetta interruzioni mascherabili prima del termine dell'esecuzione di quella successiva ad EI e che inoltre quest'ultima istruzione non può essere la prima del programma che deve

essere eseguito. Il programma principale e la subroutine d'interrupt, supposta caricata a partire dalla locazione di memoria 0400, diventano:

Programma principale		Subroutine d'interrupt	
P_1 0100	LD D,00	0400	LD B, AA
	. . . . .		LD C, 88
	. . . . .		AND B
	. . . . .		NOP
	. . . . .		LD HL, 0077
	SET 5, E	FB	EI
3E C3	LD A, C3	ED 4D	RETI
32 18 00	LD (0018), A		
FD 21 00 04	LD IY, 0400		
FD 22 19 00	LD (0019), IY		
FB	EI		
ED 46	IM 0		
C3 00 01	JP P_1		

Le istruzioni entro la parentesi hanno il compito di caricare una istruzione di salto incondizionato all'indirizzo 0400 dove è allocata la subroutine d'interrupt. Ciò si è reso necessario in quanto tale subroutine, costituita da più di 10 byte, non può essere interamente caricata a partire dalla locazione di memoria 0018 senza interessare la 0020 in cui è invece possibile effettuare un'altra restart. Il funzionamento del programma è piuttosto semplice, non appena arriva una richiesta d'interruzione automaticamente la CPU effettua una restart a partire dalla locazione di memoria 0018. Trovando in questa locazione una istruzione di salto incondizionato il program counter si porta all'indirizzo 0400 e la CPU effettua la subroutine d'interruzione.

Una volta eseguito il programma principale, si può controllare come i registri B, C ed HL contengono, se la richiesta d'interrupt è avvenuta durante l'esecuzione del programma principale (in questo caso ciò accade sempre perché il programma principale è in loop per effetto della istruzione di salto incondizionato), rispettivamente AA, 88 e 0077.

Un possibile dispositivo hardware per l'interruzione in modo 0 è mostrato in fig. 21 (*).

Il funzionamento del circuito di fig. 21 è il seguente:
non appena viene effettuata una richiesta d'interruzione (ciò può essere fatto portando per un breve istante l'ingresso $\overline{INT}$ della CPU, normal-

(*) Il periferico può essere simulato, per esempio, da otto interruttori (fig. 22) in grado di collegare gli ingressi dei tri-state a massa o a + 5 V così da fornire il vettore d'interruzione desiderato.

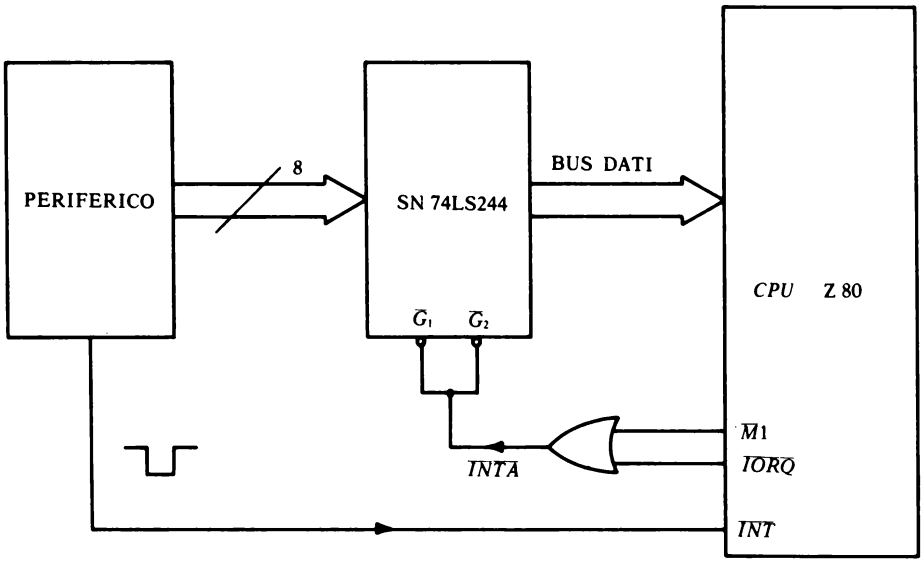


FIG. 21. - Interruzione in modo 0 con vettore d'interrupt ad un byte (istruzione di restart).

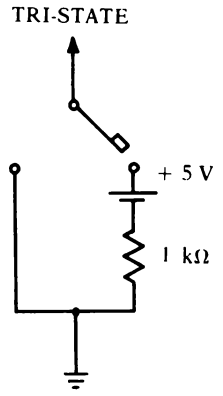


FIG. 22.

mente alto, al livello logico basso per mezzo di un pulsante munito di circuito di antirimbalzo) la $\overline{CP}$ precedentemente abilitata per via software riconosce l'interruzione (fig. 17) ed $\overline{IORQ}$ ed $\overline{M1}$, normalmente alti, si portano per un certo tempo simultaneamente al livello logico basso abilitando il porto d'ingresso SN 74LS244 (buffer tri-state) al trasferimento del vettore d'interruzione sul bus dati. A questo punto la CPU interpreta la istruzione letta come quella di restart e la esegue.

È chiaro che il vettore d'interruzione deve essere presente agli ingressi del buffer tri-state fino all'arrivo del segnale di abilitazione $\overline{INTA}$ (segnale di riconoscimento dell'interruzione) affinché l'interruzione possa essere eseguita.

Una interruzione in modo 0 può essere generata (fig. 23) abbastanza semplicemente utilizzando il componente LSI 8212 della INTEL (porto d'ingresso / uscita ad otto bit tri-state) le cui caratteristiche sono riportate nelle fig. 24 e 25.

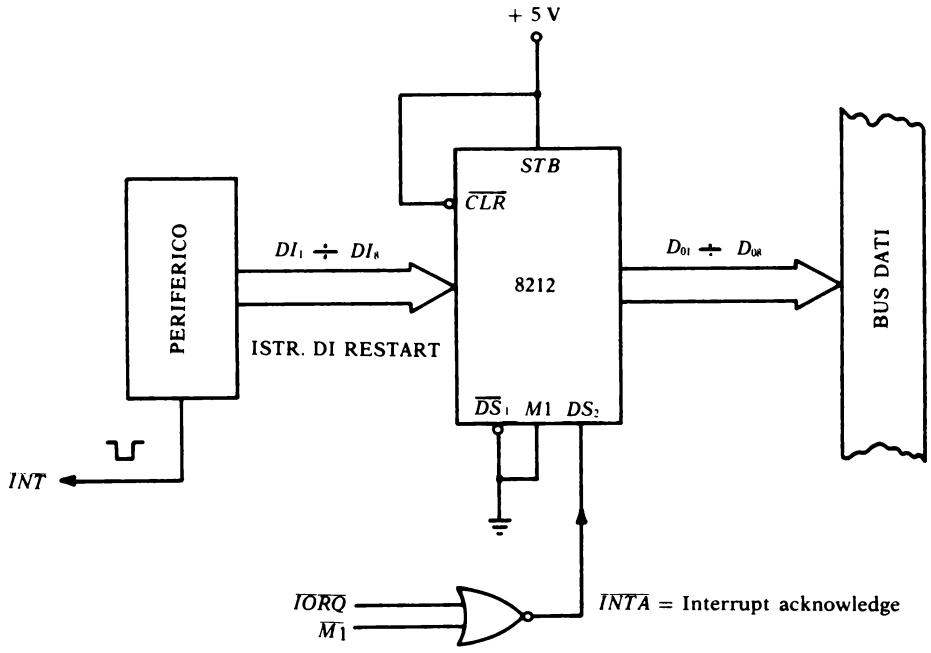


FIG. 23. - Interruzione in modo 0 mediante l'8212 (INTEL).

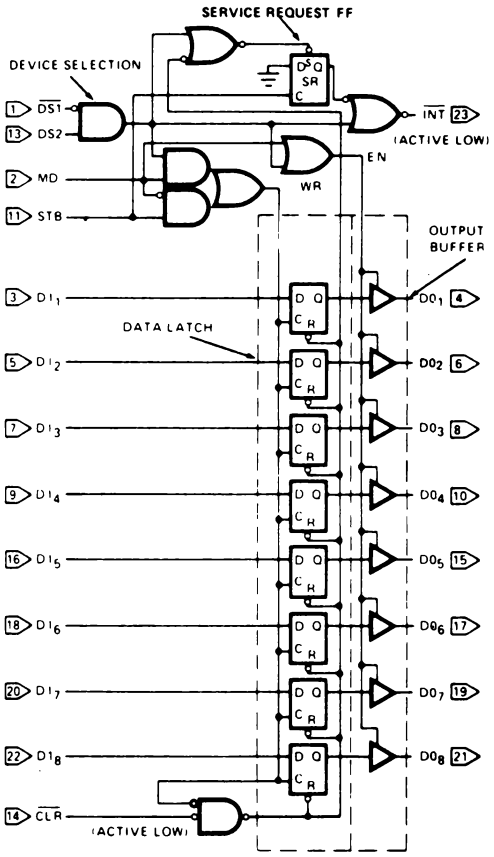
Il funzionamento dello schema di fig. 23 è il seguente:
normalmente le uscite del porto si trovano in uno stato di alta impedenza (DS2 = 0, dispositivo disabilitato). Non appena però viene effettuata una richiesta di interruzione tramite l'attivazione del piedino $\overline{INT}$, se la CPU è stata opportunamente programmata per eseguire l'interrupt in modo 0, la CPU invia un segnale di riconoscimento dell'interruzione che abilitando l'8212 (DS2 = 1) permette il trasferimento dell'istruzione di restart dal periferico al bus dei dati del sistema.

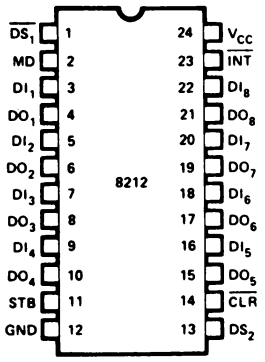
8212
8-BIT INPUT/OUTPUT PORT

- **Fully Parallel 8-Bit Data Register and Buffer**
- **Service Request Flip-Flop for Interrupt Generation**
- **Low Input Load Current — .25mA Max.**
- **Three State Outputs**
- **Outputs Sink 15mA**
- **3.65V Output High Voltage for Direct Interface to 8008, 8080A, or 8085A CPU**
- **Asynchronous Register Clear**
- **Replaces Buffers, Latches and Multiplexers in Microcomputer Systems**
- **Reduces System Package Count**

The 8212 input/output port consists of an 8-bit latch with 3-state output buffers along with control and device selection logic. Also included is a service request flip-flop for the generation and control of interrupts to the microprocessor.

The device is multimode in nature. It can be used to implement latches, gated buffers or multiplexers. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with this device.





DI_1-DI_8	DATA IN
DO_1-DO_8	DATA OUT
$\overline{DS_1}$-DS_2	DEVICE SELECT
MD	MODE
STB	STROBE
INT	INTERRUPT (ACTIVE LOW)
CLR	CLEAR (ACTIVE LOW)

FIG. 24. - Diagramma logico e piedinatura dell'8212 (INTEL).

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias Plastic ..... 0°C to +70°C
Storage Temperature ..... -65°C to +160°C
All Output or Supply Voltages ..... -0.5 to +7 Volts
All Input Voltages ..... -1.0 to 5.5 Volts
Output Currents 100mA

**NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS (T_A=0°C to +75°C, V_{CC}= +5V ± 5%)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
I_F	Input Load Current, ACK, DS_2, CR, DI_1-DI_8 Inputs			-.25	mA	V_F = .45V
I_F	Input Load Current MD Input			-.75	mA	V_F = .45V
I_F	Input Load Current DS_1 Input			-1.0	mA	V_F = .45V
I_R	Input Leakage Current, ACK, DS, CR, DI_1-DI_8 Inputs			10	μA	$V_R \leq V_{CC}$
I_R	Input Leakage Current MO Input			30	μA	$V_R \leq V_{CC}$
I_R	Input Leakage Current DS_1 Input			40	μA	$V_R \leq V_{CC}$
V_C	Input Forward Voltage Clamp			-1	V	I_C = -5mA
V_{IL}	Input "Low" Voltage			.85	V	
V_{IH}	Input "High" Voltage	2.0			V	
V_{OL}	Output "Low" Voltage			.45	V	I_{OL} = 15mA
V_{OH}	Output "High" Voltage	3.65	4.0		V	I_{OH} = -1mA
I_{SC}	Short Circuit Output Current	-15		-75	mA	V_O = 0V, V_{CC} = 5V
$\|I_O\|$	Output Leakage Current High Impedance State			20	μA	V_O = .45V/5.25V
I_{CC}	Power Supply Current		90	130	mA	

A.C. CHARACTERISTICS (T_A = 0°C to +70°C, V_{CC} = +5V ± 5%)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
t_{PW}	Pulse Width	30			ns	
t_{PD}	Data to Output Delay			30	ns	Note 1
t_{WE}	Write Enable to Output Delay			40	ns	Note 1
t_{SET}	Data Set Up Time	15			ns	
t_H	Data Hold Time	20			ns	
t_R	Reset to Output Delay			40	ns	Note 1
t_S	Set to Output Delay			30	ns	Note 1
t_E	Output Enable/Disable Time			45	ns	Note 1
t_C	Clear to Output Delay			55	ns	Note 1

FIG. 25. - Caratteristiche elettriche dell'8212 (INTEL).

Modo 1

Come si è già accennato nelle pagine precedenti il modo di interruzione 1 è il più semplice fra i tre previsti per lo Z 80. Come infatti si può osservare dalla fig. 26 per ottenere una interruzione è sufficiente inviare all'ingresso $\overline{INT}$ della CPU un impulso durante l'esecuzione del programma principale.

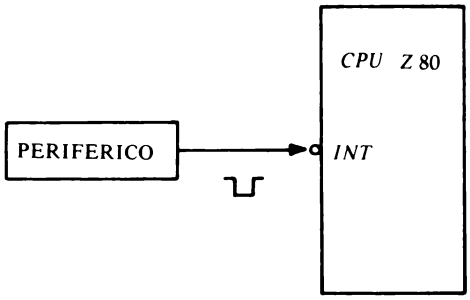


FIG. 26. - Interruzione in modo 1.

Supponendo che il programma principale e la subroutine d'interrupt siano ancora quelli del modo 0, affinché l'interruzione in modo 1 possa essere gestita è sufficiente sostituire l'istruzione IM 0 con l'istruzione IM 1 (interrupt mode 1). Inoltre poiché in questo modo d'interruzione la CPU effettua automaticamente una restart dalla locazione di memoria 0038_H occorre modificare l'indirizzo a cui deve essere caricata l'istruzione di chiamata (fig. 27).

Programma principale		Subroutine d'interrupt	
P_1 .0100	LD D, 00	0400	LD B, AA
	. . . . .		. . . . .
	. . . . .		. . . . .
	LD A, C3		EI
32 38 00	LD (0038), A		RETI
	LD IY, 0400		
FD 22 39 00	LD (0039), IY		
	EI		
ED 56	IM 1		
	JP P_1		

FIG. 27. - Interruzione in modo 1.

Modo 2

Come si è già visto nel parag. 3 per la gestione dell'interruzione nel modo 2 occorre caricare il registro I per poter puntare ad una tabella dove sono contenuti gli indirizzi a partire da cui sono caricate le subroutine d'interrupt. Così per esempio nel caso che la solita subroutine d'interrupt sia stata caricata a partire dalla locazione di memoria 0400, affinché possa essere eseguita occorre impostare nel dispositivo periferico il vettore di interruzione 00 e caricare il registro I con 04 in modo da poter formare l'indirizzo 0400. Il programma principale di pag. 148 per la gestione dell'interruzione in modo 2 diventa allora quello di fig. 28.

Programma principale

```
P1 0100          LD D, 00
                 . . . . .
                 . . . . .
                 SET 5, E
                 LD A, 04
                 LD I, A
                 EI
                 IM 2
                 JP P1
```

FIG. 28. - Interruzione in modo 2.

Il dispositivo hardware in grado di gestire l'interruzione in modo 2 è identico a quello di modo 0 di fig. 21. L'unica differenza consiste nel diverso vettore d'interruzione che occorre impostare nel dispositivo periferico: 00 anziché DF, così da puntare alla locazione di memoria 0400 nella quale è stato precedentemente caricato l'indirizzo della subroutine voluta (fig. 18). Il più delle volte accade che durante la subroutine d'interrupt debbono essere salvati i contenuti dei registri del programma principale, ciò può essere fatto mediante le istruzioni di PUSH e POP, come mostrato in fig. 29.

Subroutine d'interrupt

```
PUSH BC
PUSH DE
PUSH HL
PUSH AF
. . . .
. . . .
. . . .
POP AF
POP HL
POP DE
POP BC
EI
RETI
```

FIG. 29.

Modo non mascherabile

In questo modo la sola istruzione da inserire è quella di RET N. Tale istruzione, collocata alla fine della routine di servizio dell'interrupt, permette il rientro incondizionato al programma principale.

5. Z 80 PIO (Parallel Input/Output Interface)

Il principale dispositivo usato per la trasmissione di segnali e dati tra un sistema a microcomputer, con unità centrale di elaborazione lo Z 80, e la logica esterna è il PIO Z 80. Il PIO Z 80 è un circuito integrato a larga scala di integrazione a tecnologia NM0S programmabile mediante il quale è possibile realizzare una interfaccia, compatibile TTL, tra la CPU ed i dispositivi periferici che richiedono un trasferimento parallelo di dati. La CPU può programmare il PIO ad operare in diversi modi e questo permette di interfacciare, senza bisogno di particolari circuiti esterni, numerosi dispositivi periferici quali ad esempio: tastiere, stampanti, programmatori di PR0M etc.

Architettura interna del PIO

In fig. 30 è mostrato lo schema a blocchi del PIO Z 80. Esso è costituito da due porte bidirezionali A e B, ad otto bit indipendenti utilizzate per realizzare una interfaccia diretta con dispositivi periferici. Inoltre tale schema comprende l'interfaccia verso il bus della CPU, il controllo dell'interruzione e la logica interna di controllo che ha il compito di sincronizzare il bus dei dati della CPU con le porte A e B. Ciascuna delle porte come si vedrà in seguito può operare in tre modi programmabili diversi oltre ad un quarto modo permesso solo alla porta A. Programmabile è anche la risposta alla interruzione.

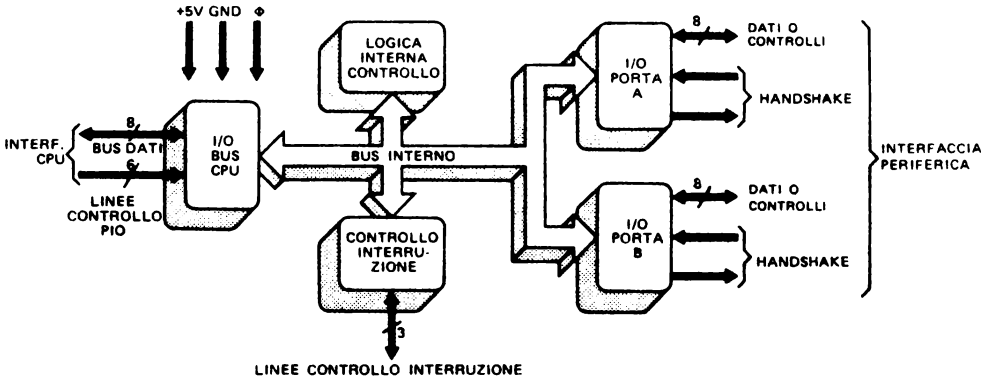


FIG. 30. - Schema a blocchi del PIO Z 80 (dal manuale tecnico SGS).

Struttura interna di una porta

In fig. 31 è riportato lo schema a blocchi di una porta del PIO. La logica di ciascuna delle due porte è realizzata mediante sei registri e da una logica di controllo di tipo *handshake* (*).

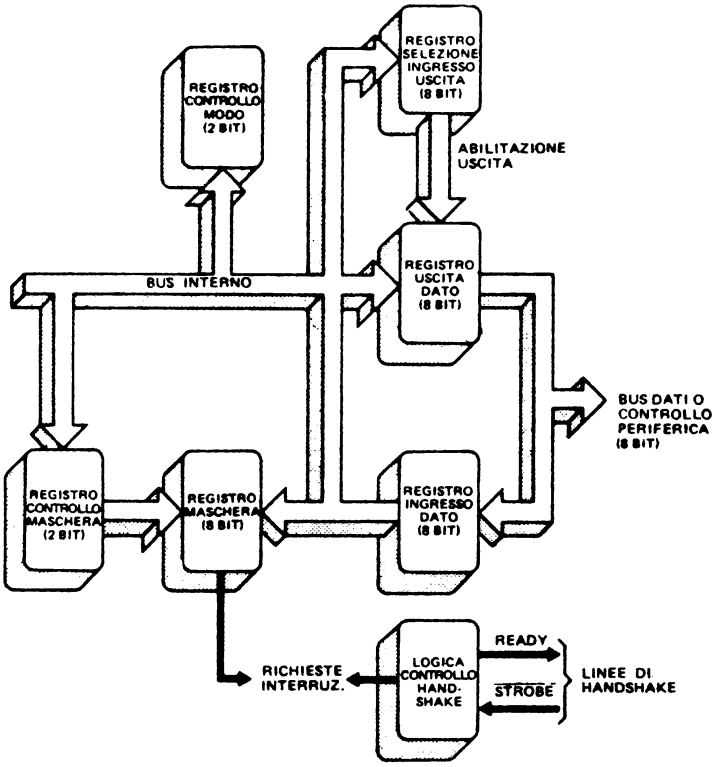


FIG. 31. - Schema a blocchi di una porta di I/O del PIO Z 80 (dal manuale tecnico SGS).

Descrizione dei registri:

Il registro di controllo modo (2 bit) permette alla CUP di controllare i modi di operazione della porta che sono:
modo 0 (uscita)
modo 1 (ingresso)
modo 2 (bidirezionale)
modo 3 (controllo dei singoli bit)

(*) *Handshake* (stretta di mano). Con questo termine si intende un processo per cui un segnale di riconoscimento è spedito indietro quale risposta al ricevimento di informazioni.

Il registro maschera ad otto bit ed il registro di selezione ingresso/uscita sono usati nel modo di controllo dei singoli bit, cosicché in questo modo ognuno degli otto pin della porta può essere programmato come ingresso o come uscita a seconda dello stato del registro di selezione.

Il registro maschera a due bit permette alla CPU di specificare quale debba considerarsi lo stato attivo (basso o alto) di ogni dispositivo periferico.

Il registro dati d'ingresso ad otto bit permette il trasferimento dei dati dal periferico alla CPU.

Il registro dati di uscita ad otto bit è utilizzato per trasferire i dati dalla CPU al periferico.

Descrizione dei segnali del PIO

In fig. 32 è mostrata la configurazione dei piedini del PIO Z 80, la loro funzione è la seguente:

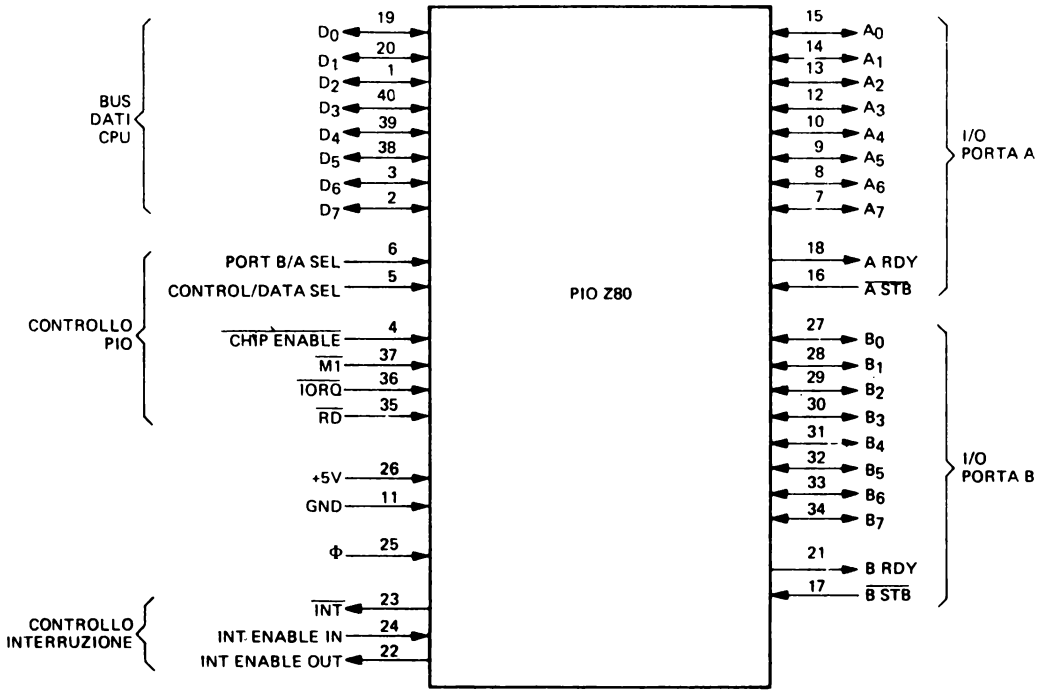


FIG. 32. - Piedinatura del PIO Z 80 (dal manuale tecnico SGS).

$D_0 \div D_7$
Bus dei dati bidirezionale tri-state della CPU utilizzato per trasferire i dati e i comandi tra la CPU ed il PIO.

B/A Sel (Port B/A Select)
Selezione porta B/A. Questo ingresso permette alla CPU di scegliere la porta con cui comunicare e a seconda del suo stato seleziona una delle due porte (livello basso selezione di A, livello alto selezione di B).

C/D Sel (Control/Data Select)
Selezione controllo dato (attivo alto). Questo ingresso determina l'esatta interpretazione della parola presente sul bus dei dati.

$\overline{CE}$ (Chip Enable)
Abilitazione del chip, ingresso attivo basso.

Φ (System Clock)
Il PIO Z 80 usa il clock standard del sistema Z 80 per sincronizzare alcuni segnali interni, è un clock ad una sola fase.

$\overline{M1}$ (Machine Cycle)
Segnale di ciclo macchina 1 generato dalla CPU. Ingresso attivo basso, agisce come segnale di sincronismo.

$\overline{IORQ}$ (Input/Output Request)
Richiesta d'ingresso / uscita generata dalla CPU (attiva bassa).

$\overline{RD}$ (Read Cycle)
Segnale di ciclo lettura generato dalla CPU. Se tale ingresso è attivo (livello logico basso) vuol dire che è in corso una operazione di lettura in memoria oppure una operazione di lettura in un dispositivo I/O.

IEI (Interrupt Enable In)
Ingresso di abilitazione dell'interruzione (attivo alto). È un segnale utilizzato per formare un controllo daisy-chain dell'interrupt. Un livello logico alto su questo pin indica che la CPU non sta eseguendo la routine d'interruzione di nessun altro dispositivo a priorità più elevata.

IEO (Interrupt Enable Out)
Uscita di abilitazione dell'interruzione attiva alta. Questo pin si porta al livello logico alto quando IEI è alto ed il microprocessore non sta servendo una interruzione generata dal PIO.
Questo impedisce l'esecuzione d'interrupt a priorità più bassa di quella che la CPU sta eseguendo.

$\overline{INT}$ (Interrupt Request)
Richiesta d'interruzione, uscita attiva bassa.

$A_0 \div A_7$
Bus dei dati bidirezionale tri-state della porta A.

$B_0 \div B_7$
Bus dei dati bidirezionale tri-state della porta B.

$\overline{A\ STB}$, $\overline{B\ STB}$ (Port A Strobe, Port B Strobe)
Ingressi (attivi bassi) che determinano l'esecuzione della operazione richiesta.

A RDY, B RDY (Register A Ready, Register B Ready)
Uscite attive alte che segnalano che le porte corrispondenti sono pronte a compiere l'operazione richiesta.

6. Programmazione del PIO Z 80

Non appena viene fornita la tensione di alimentazione, analogamente a quanto avviene per la CPU, il PIO si pone automaticamente in uno stato di azzeramento o reset. In particolare durante questo stato sono eseguite le seguenti operazioni:
— I registri di maschera delle porte A e B vengono azzerati.
— Le linee del bus dei dati delle porte sono messe in uno stato di alta impedenza ed i segnali di RDY della logica di controllo (handshake) sono disattivati.
— Viene selezionato automaticamente il modo 1 di operazione mentre i registri che contengono il vettore d'interruzione non vengono azzerati.
— I flip flop di abilitazione dell'interruzione delle porte sono disabilitati mentre i registri di uscita delle stesse porte vengono azzerati.

È comunque possibile generare un reset del PIO senza per questo dover togliere la tensione di alimentazione, ciò può essere fatto abbastanza semplicemente inviando un segnale basso al pin $\overline{M1}$ con $\overline{RD}$ e $\overline{IORQ}$ al livello logico alto.

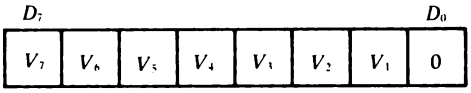
Il PIO una volta entrato in uno stato di reset vi resta fino all'arrivo di una parola di controllo inviata dalla CPU.

La programmazione del PIO Z 80 in generale può essere pensata suddivisa in tre fasi principali:
1) Caricamento del vettore d'interruzione.
2) Definizione del modo di funzionamento.
3) Definizione del controllo delle interruzioni.

Caricamento del vettore d'interruzione

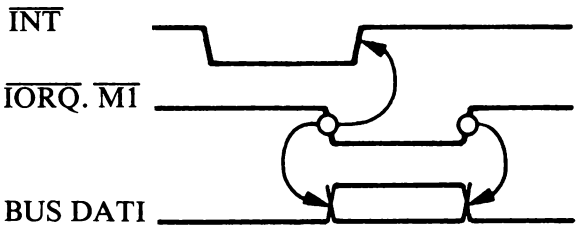
Si è già visto nei paragrafi precedenti come agiscono i tre modi possibili d'interruzione nello Z 80. Il PIO Z 80 è stato progettato per operare soltanto nel modo 2 d'interruzione, modo questo che richiede una generazione di un vettore d'interruzione da parte del periferico per formare l'indirizzo della subroutine di servizio dell'interrupt richiesta. Ogni porta del PIO possiede un vettore d'interruzione indipendente che può essere caricato nel PIO scrivendo nella porta scelta una parola di controllo del seguente formato:

FIG. 33. -



Il bit meno significativo della parola di controllo messo a zero sta ad indicare che si è in presenza di un vettore d'interruzione. D_0 viene usato come bit di flag che quando è a 0 permette il caricamento nel registro vettore dei bit da V_7 a V_1.

Durante il ciclo di riconoscimento dell'interruzione, il vettore corrispondente alla porta del PIO che ha richiesto l'interrupt può essere caricato tramite il bus dati nella CPU secondo il formato visto in fig. 33. In definitiva ecco ciò che accade non appena il PIO richiede l'interruzione mediante l'attivazione del piedino $\overline{INT}$ (fig. 34).



Segnale di richiesta d'interrupt generato dal PIO

Eventuale risposta all'interrupt da parte della CPU.

Inserimento sul bus dati della CPU del vettore di interruzione.

FIG. 34. -

Come si è già accennato nel parag. 4 il PIO Z 80 è provvisto di un controllo di priorità del tipo daisy chain. Per realizzare tale tipo di controllo basta collegare l'ingresso IEI del PIO a priorità più elevata a + 5 volt e l'uscita IE0 al pin IEI del PIO successivo secondo lo schema di fig. 35.

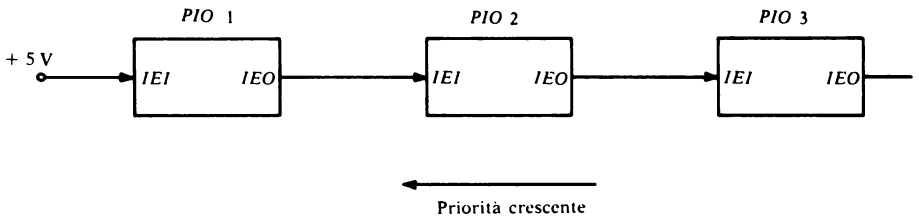


FIG. 35. - Controllo daisy chain in PIO collegati in cascata.

Tutto ciò avviene senza l'ausilio di una logica esterna soltanto se il numero dei PI0 collegati in daisy chain non supera le quattro unità. Questa limitazione è dovuta al tempo finito di propagazione dello stato di abilitazione dell'interruzione lungo la catena. Infatti facendo riferimento alla fig. 36 si può notare come tale propagazione debba poter avvenire in un tempo compreso tra l'inizio di $\overline{M1}$ e l'inizio di $\overline{IORQ}$. D'altra parte non potendo variare lo stato di abilitazione dell'interrupt durante $\overline{M1}$ il vettore d'interruzione che viene letto dalla CPU verrà presentato dal PI0 a più alta priorità fra quelli che hanno effettuato la richiesta d'interrupt.

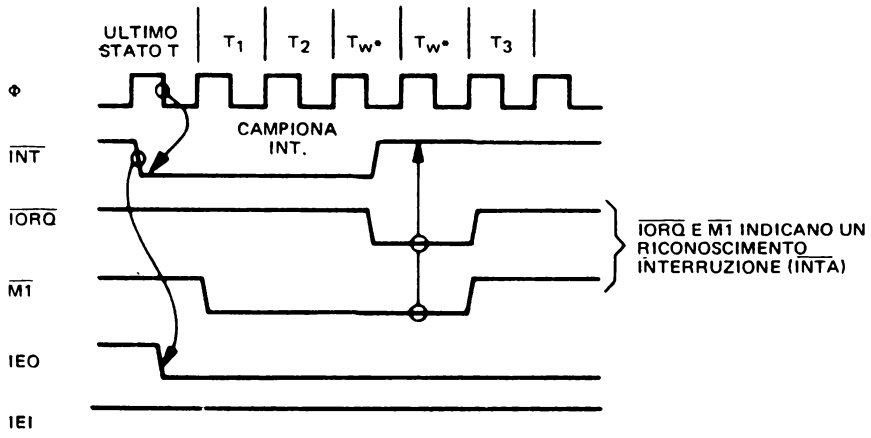


FIG. 36. - Ciclo di riconoscimento delle interruzioni (dal manuale tecnico SGS).

Definizione del modo di funzionamento

Come si è già accennato precedentemente i quattro modi di funzionamento del PI0 sono:
- Modo 0 o modo d'uscita
- Modo 1 o modo d'ingresso
- Modo 2 o modo bidirezionale (accessibile solo alla porta A)
- Modo 3 o modo di controllo di bit.

Il modo di funzionamento può essere scelto dall'operatore scrivendo nel PI0 una opportuna parola di controllo (fig. 37).

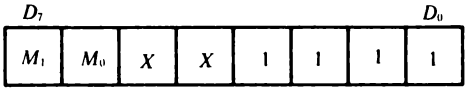


FIG. 37. - Formato della parola di controllo.

I bit D_7 e D_6 definiscono il modo di operazione del PI0 secondo la tabella di fig. 38.

M_1	M_0	Modo
0	0	Uscita
0	1	Ingresso
1	0	Bidirezionale
1	1	Controllo di bit

FIG. 38.

Quando viene selezionato il modo di controllo di bit la successiva parola a quella di controllo deve definire quali fra le linee del bus dei dati debbono essere considerate quelle di input e quali quelle di output. Riferendosi alla fig. 39 se I/0 = 1 la corrispondente linea del bus dei dati sarà usata come ingresso, se I/0 = 0 come uscita

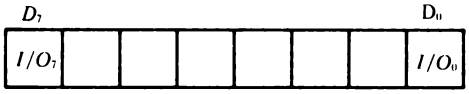


FIG. 39.

Modo di uscita o modo 0

Il modo di uscita permette ad una delle due porte od a entrambe di essere usate come mezzi di trasferimento dei dati verso il dispositivo periferico. In fig. 40 è mostrato il diagramma di temporizzazione del modo 0.

Quando la CPU esegue una istruzione d'uscita essa genera dei segnali di controllo che il P10 combina con il suo segnale interno di scrittura $\overline{WR}$*. Attraverso questo segnale il dato presente sul bus dei dati della CPU viene caricato nel registro di uscita della porta scelta.

Per indicare che il dato è stabilizzato e disponibile al periferico viene attivato il segnale RDY. Questo segnale rimane alto fino a che il dispositivo periferico non invia un segnale basso all'ingresso di strobe $\overline{STB}$.

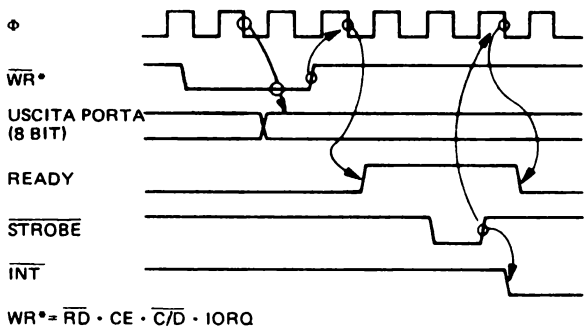


FIG. 40. - Temporizzazioni del modo 0 (dal manuale tecnico SGS).

Nella successiva transizione del clock dal livello alto al livello basso il segnale di RDY ritorna basso. La transizione dal livello basso al livello alto di STB genera una richiesta di interruzione, se l'interruzione è stata abilitata.

Modo d'ingresso o modo 1

In fig. 41 è riportato il diagramma di temporizzazione per il modo 1.

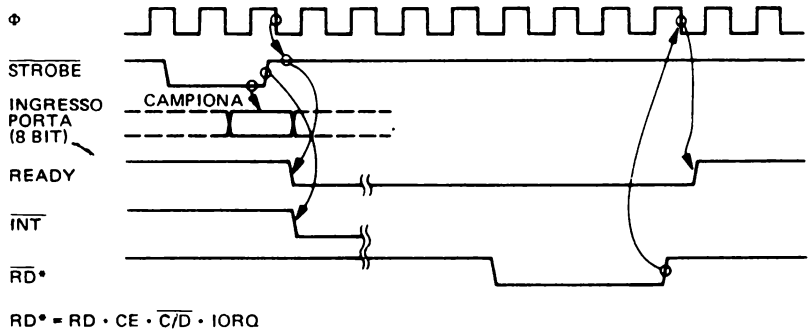


FIG. 41. - Temporizzazioni del modo 1 (dal manuale tecnico SGS).

In questo caso il periferico inizia un ciclo d'ingresso inviando un segnale basso di strobe STB. Questo segnale permette di caricare il dato nel registro d'ingresso della porta prescelta. Durante il fronte di salita del segnale STB viene inviata una richiesta d'interruzione se l'interruzione è stata abilitata. Durante il fronte di discesa dell'impulso di clock che segue l'ingresso alto STB, il segnale di ready (RDY) diventa inattivo per indicare alla logica esterna che nessun dato può essere ricevuto.

RDY rimane basso fino a che la CPU non ha letto il dato, subito dopo si porta al livello logico alto.

Modo bidirezionale o modo 2

Nel modo di controllo bidirezionale le quattro linee di controllo handshake sono utilizzate per trasferire i dati in modo bidirezionale mediante la porta A. La porta B deve essere predisposta al funzionamento in modo 3. In fig. 42 è mostrato il diagramma di temporizzazione del modo 2.

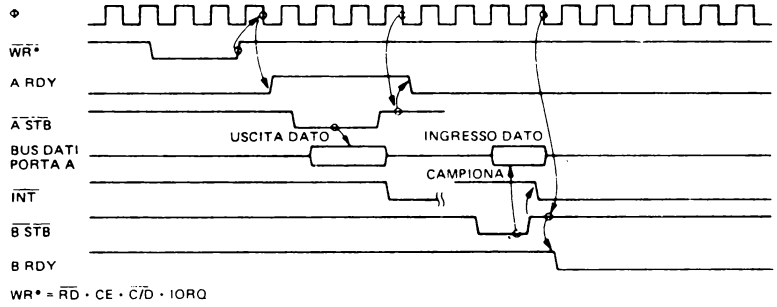


FIG. 42. - Temporizzazioni del modo 2 (dal manuale tecnico SGS).

L'unica caratteristica di questo modo è che i dati sono in uscita sulla porta A soltanto durante l'attivazione dello strobe ($\overline{A\ STB}$). Ciò si rende necessario poiché in questo modo i pin della porta A debbono essere in grado di ricevere dati in ingresso non appena completata l'operazione di uscita.

Modo di controllo di bit o modo 3

In questo modo non vengono usati segnali di controllo o handshake. È possibile eseguire qualunque operazione di lettura o scrittura in qualsiasi momento. Durante la fase di scrittura la temporizzazione è uguale a quella del modo 0. In fase di lettura il dato inviato alla CPU è costituito dalla combinazione del registro d'ingresso e di quello d'uscita in base al contenuto del registro di selezione ingresso-uscita (fig. 43).

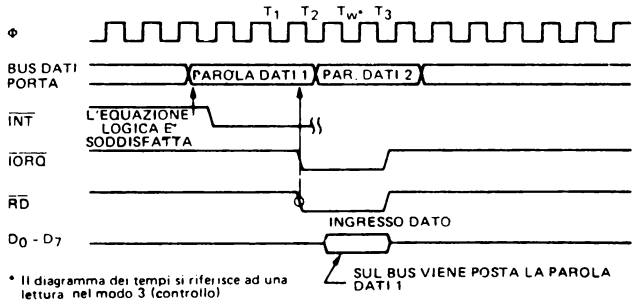


FIG. 43. - Temporizzazioni del modo 3 (dal manuale tecnico SGS).

Definizione del controllo delle interruzioni

Il PI0 possiede una parola di controllo d'interruzione per ogni porta (A e B) che determina sotto quali condizioni una richiesta d'interruzione sarà inviata alla CPU. La parola di controllo dell'interruzione ha il seguente formato:

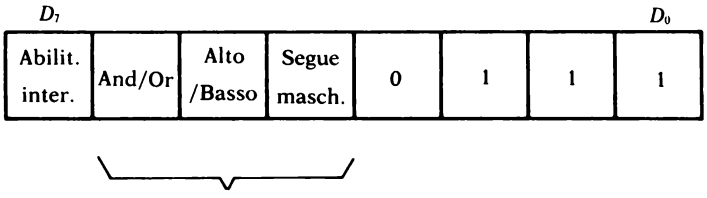


FIG. 44. - Parola di controllo dell'interruzione.

Gli ultimi quattro bit della parola stanno ad indicare che essa è una parola di controllo dell'interruzione. Il bit D_6 definisce l'operazione logica che deve essere eseguita sul dato presente nella porta. Se D_6 è uguale ad 1 viene definita una funzione logica And se invece D_6 è uguale a 0 viene selezionata la funzione logica Or. Nel primo caso tutti i bit selezionati della porta debbono essere portati al livello logico alto affinché possa essere generata una richiesta di interruzione; nel caso della funzione Or viene generata una interruzione anche se un solo bit si è portato nello stato attivo. Il bit D_5 definisce quale è lo stato attivo delle linee del bus dei dati della porta.

Se D_5, vale 1, delle linee dei dati della porta è considerato attivo lo stato alto, in caso contrario ($D_5 = 0$) lo stato basso. Se il bit D_4 vale 1 la successiva parola di controllo inviata al PI0 definisce una maschera del seguente formato:

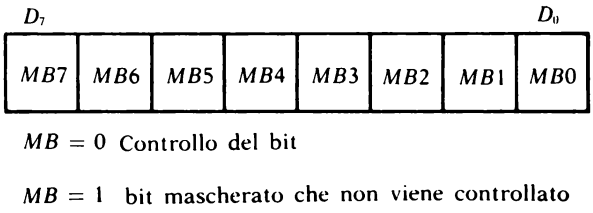
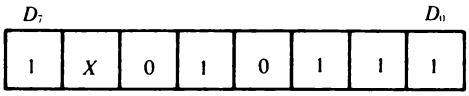


FIG. 45. - Parola di maschera.

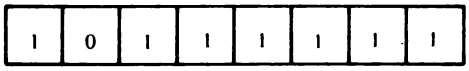
Soltanto quelle linee della porta il cui bit di maschera è uguale a zero saranno controllate per generare una interruzione. Per esempio se la parola di controllo ha il seguente formato:

FIG. 46.



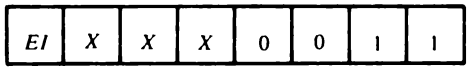
il formato della parola di maschera che permette soltanto il controllo del bit D_6 vale:

FIG. 47.



Il bit D_6 della parola di controllo è posto indifferentemente a 0 od a 1 in quanto dovendo controllare soltanto un segnale non ha importanza specificare la funzione logica. Inoltre il flip-flop di abilitazione di una porta può essere settato o resettato senza modificare la parte restante della parola di controllo dell'interruzione utilizzando il seguente comando:

FIG. 48.



Comunque, indipendentemente dal modo di funzionamento del PI0, se il bit D_4 della parola di controllo viene posto ad 1 si ha sempre la cancellazione di qualsiasi interruzione pendente.

Indirizzamento del PI0

Ogni Z 80 PI0 possiede quattro porti di accesso indirizzabili. I tre pin del PI0 $\overline{CE}$, B/A e C/D sono utilizzati per selezionare il dispositivo e le sue porte secondo la seguente tabella:

$\overline{CE}$	C/D	B/A	Locazione selezionata
0	0	0	Porta A parte dati
0	1	0	Porta A parte controllo
0	0	1	Porta B parte dati
0	1	1	Porta B parte controllo
1	X	X	PI0 non selezionato

FIG. 49.

Determinazione dell'indirizzo di I/0

Quando la CPU esegue una istruzione d'ingresso o di uscita è possibile selezionare un dispositivo di I/0 mediante l'assegnazione di un codice dispositivo ad otto bit formato da quelli meno significativi presenti sul bus degli indirizzi (parag. 1). Questi otto bit (da A_0 ad A_7) permettono la selezione di $2^8 = 256$ diversi dispositivi di I/0. Nel caso di due soli PI0 collegati come in fig. 50 si hanno a disposizione soltanto 8 locazioni di I/0 indirizzabili e di conseguenza sono sufficienti ad esempio gli ultimi tre bit del bus degli indirizzi per poterle selezionare (fig. 51).

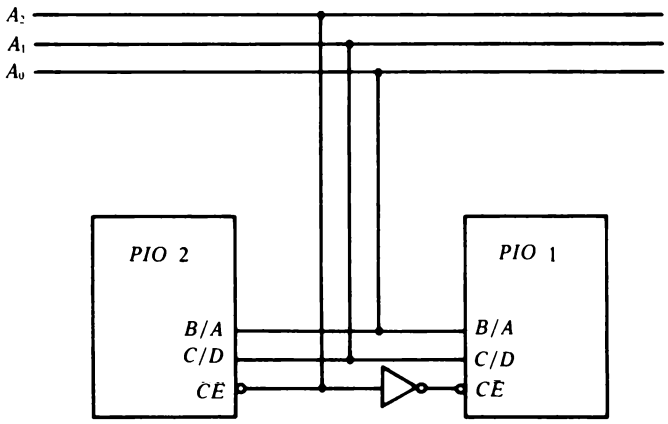
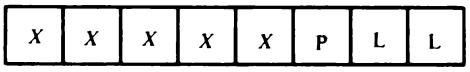


FIG. 50. - Collegamento di due PIO.



X = indifferentemente 0 od 1
P = selezione del PIO
LL = selezione della locazione

FIG. 51. - Contenuto della parte bassa del bus degli indirizzi durante una operazione di I/O.

In particolare gli ultimi due bit selezionano le locazioni di I/0 mentre il bit A_2 seleziona uno dei due PI0 secondo la seguente tabella:

Indirizzo A_7	A_0	PI0 e porta selezionati
XXXXX	000	PI0 2 porta A parte dati
XXXXX	010	PI0 2 porta A parte controllo
XXXXX	001	PI0 2 porta B parte dati
XXXXX	011	PI0 2 porta B parte controllo
XXXXX	100	PI0 1 porta A parte dati
XXXXX	110	PI0 1 porta A parte controllo
XXXXX	101	PI0 1 porta B parte dati
XXXXX	111	PI0 1 porta B parte controllo

FIG. 52.

7. Esempi di programmazione del PIO nei vari modi di funzionamento

In questo paragrafo saranno esaminate tutte le istruzioni necessarie per la programmazione del PIO nei vari modi di funzionamento. Per la comprensione dei singoli programmi il lettore tenga presente quanto descritto nei paragrafi precedenti riguardo ai modi di funzionamento del PIO ed al modo 2 di interruzione. Gli indirizzi delle porte del PIO sono stati ricavati dalla tabella di fig. 52. Effettuando, per esempio, una decodifica in modo che il PIO 2 venga selezionato soltanto ogni qualvolta le linee di indirizzo da A_7 ad A_3 sono uguali a 00001 ed il PIO 1 nel caso invece che le stesse linee valgono 00000.

Modo 0 o modo di uscita

```
LD A, 0F          Programma in modo 0 la porta B del PIO
OUT (0B), A
LD A, N           Carica un dato
OUT (09), A       Poni in uscita il dato sulla porta B
HALT
```

Modo 1 o modo d'ingresso

```
LD A, 4F          Programma in modo 1 la porta B del PIO
OUT (0B), A
IN A, (09)        Carica il dato nel registro d'ingresso della
                  porta B
HALT
```

Modo 0 con handshake

```
    IM 2              Predisponi l'interruzione in modo 2
    LD A, TAB H       Carica in I la parte alta dell'indirizzo puntatore di tabella
    LD I, A
    LD IX, Subr.      Carica l'indirizzo della subroutine d'interrupt in tabella.
    LD (TAB), IX
    LD A, TAB L       Carica il vettore d'interruzione nella porta B del PIO
    OUT (0B), A
    LD A, OF          Programma il PIO in modo 0 per la porta B
    OUT (0B), A
    LD A, 87          Carica la parola di controllo delle interruzioni sulla porta B del PIO
    OUT (0B), A
P1  EI
    JP P1 oppure JP Main
```

```
Main  EI                    Subr  . . . . .
      . . . . .                   . . . . .
      . . . . .                   OUT (09), A
      JP Main                     EI
                                  RETI
```

Il modo 1 con hanshake è identico al modo 0 per quanto riguarda le inizializzazioni eccetto per il byte di modo: 4F anziché 0F. Inoltre nella subroutine d'interrrupt occorre inserire l'istruzione IN A, (09) al posto di OUT (09), A.

Modo 2 con segnali di handhaske delle due porte

```
IM 2
LD, A Tabella          Carica in I la parte alta della tabella
                       (In questo esempio Tabella indica un
                       puntatore d'inizio pagina, es. 0700,
                       0800, ...., 0F00 etc.)
LD I, A

LD IX, Subr. OUT       Carica l'indirizzo di Subr OUT in ta-
                       bella
LD (Tabella + 02), IX

LD A, 02               Carica il vettore d'interruzione relati-
                       vo alla porta A
OUT (0A), A

LD IX, Subr. IN        Carica l'indirizzo di Subr IN in tabella
LD (Tabella + 04), IX

LD A, 04               Carica il vettore d'interruzione relati-
                       vo alla porta B
OUT (0B), A

LD A, 8F               Predisponi il modo 2 per la porta A
                       del PI0
OUT (0A), A

LD A, CF               Predisponi il modo 3 per la porta B
                       del PI0
OUT (0B), A
```

```
        LD A, FF                 Predisponi la maschera I/0 necessaria
                                 dopo il modo 3. In questo esempio le
                                 linee del bus dati della porta B sareb-
                                 bero selezionate come ingressi; in ef-
                                 fetti nel modo 2 i dati passano solo
                                 nella porta A.
        OUT (0B), A

        LD A, 87                 Carica la parola di controllo delle in-
                                 terruzioni sulle due porte
        OUT (0A), A
        OUT (0B), A
        JP Main

Main EI        Subr OUT     .......... Subr IN   ..........
     ..........             ..........           ..........
     ..........             OUT (08), A          IN A, (08)
     JP  Main               EI                   EI
                            RETI                 RETI
```

Una volta programmato il PI0, per generare una interruzione occorre inviare un impulso basso ad uno dei due ingressi di strobe delle due porte del PI0 (fig. 42) e precisamente in $\overline{A\ STB}$ se si vuole l'uscita di dati e in B $\overline{STB}$ se si desidera l'ingresso di dati.

Modo 3

```
        IM 2
        LD A, Tabella
        LD I, A
        LD IX, Subr 3
        LD (Tabella + 08), IX
        LD A, 08
        OUT (0A), A
        LD A, CF                 Predisponi il modo 3 per la porta A
        OUT (0A), A

        LD A, E0                 Predisponi la maschera I/0 (sono sele-
        OUT (0A), A              zionate le prime tre linee in ingresso e
                                 le altre in uscita)

        LD A, B7                 Predisponi la parola di controllo della
                                 interruzione
        OUT (0A), A
```

```
        LD A, C3            Formato della maschera di controllo
                            (sono sotto controllo le linee B5, B4, B3
                            e B2).
        OUT (0A), A         Poiché si sono considerati attivi gli 1 e
                            la funzione logica è Or si avrà una in-
                            terruzione non appena uno dei bit sot-
                            to controllo passa da 0 ad 1 (se erano
                            tutti zero).
        JP Main

Main    EI          Subr 3  . . . . .
        . . . . .           . . . . .
        . . . . .           IN A, (08)
        JP Main             . . . . .
                            . . . . .
                            OUT (08), A
                            . . . . .
                            . . . . .
                            EI
                            RETI
```

8. DMA (Direct Memory Access)

La lettura di un dato da un dispositivo periferico e la successia scrittura in memoria o viceversa, anziché passare attraverso la CPU può avvenire direttamente mediante un dispositivo di controllo capace di eseguire trasferimenti di dati fra la memoria interna del sistema ed i periferici di I/0 (fig. 53).

Questo dispositivo, a cui si dà il nome di DMA, permette una maggiore velocità di lavoro da parte del sistema. Resta inteso che durante i trasferimenti di dati mediante il DMA, l'unità centrale di elaborazione deve rimanere scollegata dal bus dei dati mediante dispositivi tri-state. Anche la famiglia Z 80 della Zilog possiede un proprio dispositivo DMA. Lo Z 80 DMA è un circuito programmabile a tecnologia NM0S in singolo chip, che permette di eseguire operazioni di accesso diretto in memoria fornendo indirizzi, segnali di controllo e di temporizzazioni necessari per effettuare il trasferimento dei dati tra due parti del sistema. Ciascuna di queste due parti può essere sia la memoria principale sia qualunque dispositivo periferico. Sono possibili quindi trasferimenti di dati tra due periferici, fra una memoria ed un periferico e fra due memorie. Il DMA Z 80 può inoltre ricercare un byte particolare in un blocco di dati con o senza il simultaneo

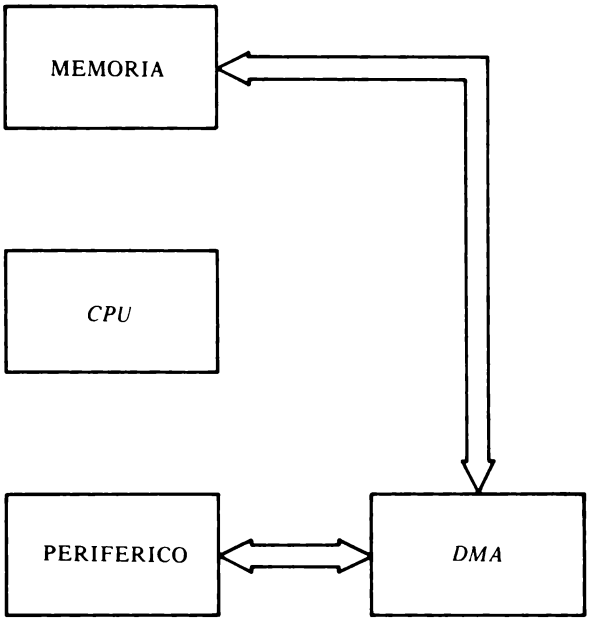


FIG. 53. -
Diretto accesso in memoria

trasferimento dei dati esaminati. Per quanto riguarda le caratteristiche tecniche e la descrizione dettagliata di questo componente così come per il *SIO Z 80 (Serial Input/Output, componente periferico capace di soddisfare una larga varietà di esigenze nella trasmissione dei dati nei sistemi a microprocessore) si rimanda il lettore alla consultazione dei manuali tecnici specifici della SGS.*

9. Controllo di un sistema industriale di processo

In fig. 54 è mostrato lo schema a blocchi di un dispositivo di controllo di un sistema industriale basato sulla CPU e PI0 Z 80.

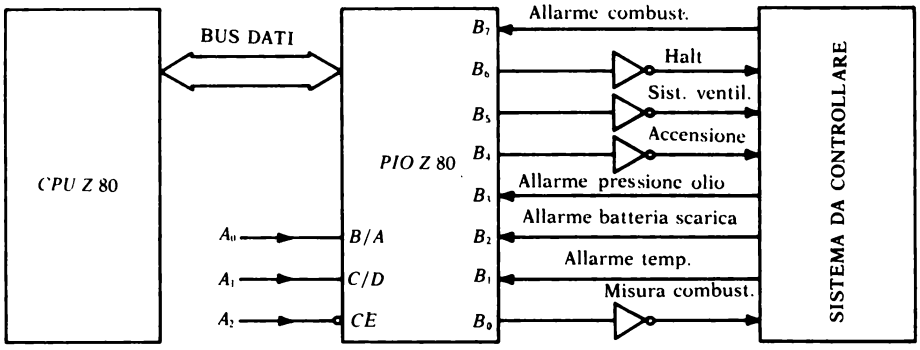


FIG. 54. - Controllo di un sistema industriale.

Se la parola di stato del sistema da controllare è quella di fig. 55 la programmazione del PI0 avviene secondo le seguenti fasi.

D_7							D_0
Allarme combustibile al minimo	Blocca il processo	Inserisci il sistema di ventilazione	Accensione	Allarme pressione olio	Allarme batteria scarica	Allarme temperatura	Misura il combust.

FIG. 55. - Parola di stato del sistema.

Si seleziona la porta (per es. B) ed il modo di funzionamento 3 del PI0 scrivendo nella porta B la relativa parola di controllo (fig. 56) tenendo presente la tabella di fig. 38.

FIG. 56. - Parola di controllo che seleziona la porta B del PIO nel modo 3.

D_7							D_0
1	1	*X*	*X*	1	1	1	1

(es. *FF*)

Stabilito che gli ingressi sono B_7, B_3, B_2 e B_1 mentre le uscite sono B_6, B_5, B_4 e B_0, ne segue la seguente parola di selezione di I/0:

FIG. 57. - Parola di selezione di I/O.

D_7							D_0
1	0	0	0	1	1	1	0

Il caricamento del vettore d'interruzione avviene secondo il modo 2 ricordando che l'ultimo bit del vettore deve essere 0 (fig. 58).

FIG. 58. - Formato del vettore d'interruzione.

D_7							D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

La parola di controllo d'interruzione da inviare alla porta B vale:

FIG. 59. - Parola di controllo d'interruzione.

D_7							D_0
1	0	1	1	0	1	1	1

mentre la parola di maschera che segue quella di controllo d'interruzione e abilita le linee B_7, B_3, B_2 e B_1, è quella di fig. 60.

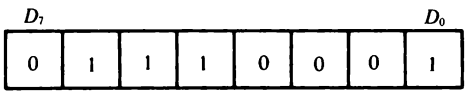


Fig. 60. - Parola di maschera.

Se ad un certo istante un fine corsa od un sensore porta al livello logico alto una delle linee B_7, B_3, B_2 o B_1, viene generata una richiesta d'interruzione. Così per esempio se ad un certo momento scatta l'allarme relativo al controllo della temperatura, l'interruzione generata dal sensore B_1 porta la CPU ad eseguire la subroutine d'interrupt, nel caso che siano state abilitate le interruzioni da programma. Una volta richiamata la subroutine di controllo della temperatura mediante una istruzione di uscita è possibile, programmando opportunamente il PI0, mettere in uscita sulla linea B_5 un segnale di comando capace di attivare il sistema di raffreddamento del dispositivo sotto controllo. La sequenza delle istruzioni che programmano la porta B del PI0 è la seguente:

```
LD A, FF          Predisponi il modo 3 per la porta B
OUT (0B), A

LD A, 8E          Definisci le linee di I/0
OUT (0B), A

LD A, Vett        Definisci il vettore d'interrupt
OUT (0B), A

LD A, B7          Definisci la parola di controllo
OUT (0B), A

LD A, 71          Definisci la parola di maschera
OUT (0B), A
```

Naturalmente il sistema di fig. 54, così come ogni altro dispositivo di controllo basato a microprocessore,per poter essere operativo necessita di altri due elementi essenziali oltre all'alimentatore quali la memoria, in cui è scritto il programma utente ed il programma monitor, ed il clock.

Un dispositivo così strutturato e riportato in fig. 61 viene conosciuto sotto il nome di *sistema minimo*.

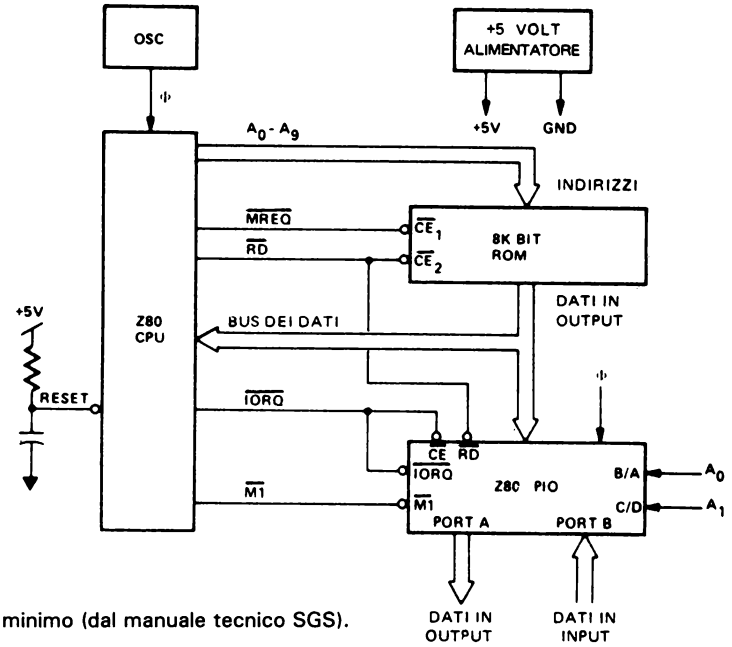


FIG. 61. - Sistema minimo (dal manuale tecnico SGS).

10. Interfacciamento di un display

Uno dei dispositivi più diffusi per la visualizzazione dei dati è quello effettuato mediante dei display. In fig. 63 è mostrato un semplice circuito di interfaccia fra un display a sette segmenti ed un microprocessore. Come porto di I/0 viene utilizzato l'integrato SN 74LS374 (Texas) la cui piedinatura e tabella della verità sono quelle di fig. 62. Per la gestione del periferico (display) viene usata una tecnica I/0 Memory Mapped.

'LS374, 'S374
FUNCTION TABLE

OUTPUT CONTROL	CLOCK	D	OUTPUT
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

See explanation of function tables on page 3-8.

SN54LS374, SN54S374 . . . J PACKAGE
SN74LS374, SN74S374 . . . J OR N PACKAGE
(TOP VIEW)

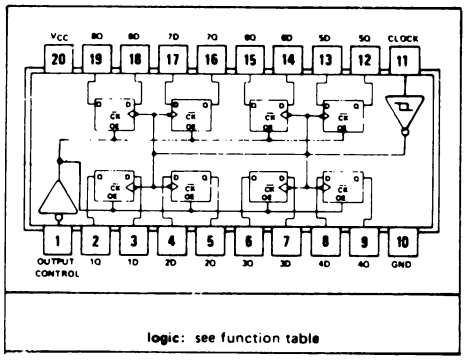


FIG. 62. -
Piedinatura e tabella della verità dell'SN 74LS374 (Texas).

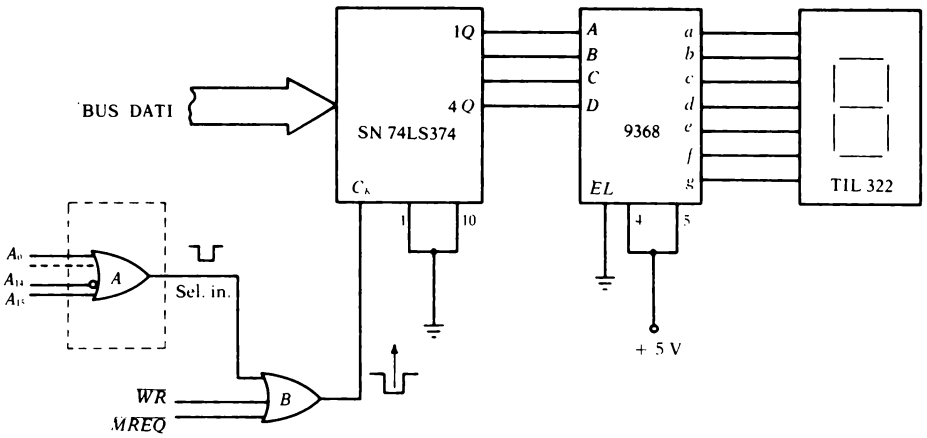


FIG. 63. - Interfaccia di un display con un microprocessore.

Il funzionamento del circuito è il seguente:
supposto di aver assegnato al porto di uscita l'indirizzo 4000, non appena vengono eseguite le istruzioni

LD A, 08
LD (4000), A

viene generato un impulso basso all'uscita della porta logica A e gli otto bit 00001000 vengono trasferiti con il fronte di salita dell'impulso dall'accumulatore all'uscita dell'SN 74LS374 (fig. 64).

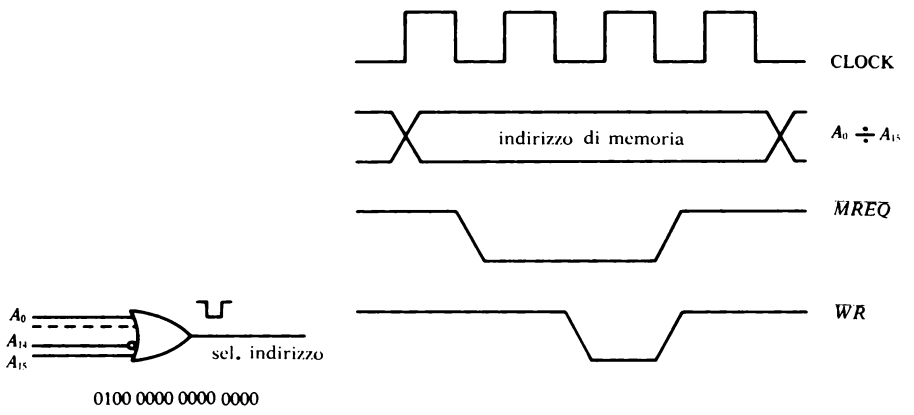
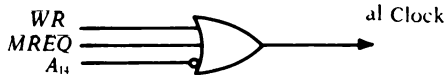


FIG. 64. - Temporizzazioni durante l'esecuzione dell'istruzione LD (nn), A.

I quattro bit meno significativi 1000 vengono decodificati dal 9368 ed il display visualizza la cifra otto. Subito dopo l'esecuzione della seconda istruzione l'uscita della porta logica A torna al livello logico alto e rimane in tale stato fino a quando non vengono eseguite di nuovo le due istruzioni di caricamento. La scrittura nel display di dati diversi può essere effettuata variando opportunamente il numero caricato nell'accumulatore. Chiaramente la decodifica d'indirizzo del porto di uscita può essere fatta abbastanza semplicemente mediante un Or a tre ingressi (fig. 65) se l'indirizzo del porto non deve occupare soltanto un byte di memoria fittizia. Infatti usando una decodifica ad un bit (A_{14}) il porto di uscita viene abilitato non soltanto per l'indirizzo 4000 ma anche per tutti quelli il cui bit A_{14} vale 1.

FIG. 65.



11. Interfaccia tra due sistemi a microprocessore

Nella comunicazione fra due sistemi a microprocessore si preferisce quanto più possibile far uso del software anziché dell'hardware, ciò comporta una possibilità di guasti minore ed una manutenzione piuttosto limitata. In fig. 66 è mostrato lo schema di un circuito di interfaccia tra due microcomputer, in cui uno si comporta da trasmettitore e l'altro da ricevitore, per uno scambio monodirezionale delle informazioni. La tecnica di comunicazione usata è del tipo *handshake*, tale tecnica consiste nell'invio di segnali particolari (ad esempio di dato pronto), detti di handshake, per permettere lo scambio dell'informazione. La lunghezza della linea deve essere limitata a qualche metro per evitare possibili riflessioni dovute alla elevata velocità del segnale di enable. L'eventuale inserimento di un latch per l'emissione del segnale di enable elimina gli inconvenienti dovuti ai tempi di propagazione nel caso dovessero presentarsi. Il trasferimento delle informazioni avviene in un interrupt, ossia durante la gestione dell'interrupt si ha l'immagazzinamento dei dati. Il microprocessore ricevente una volta letta l'informazione deve dare la conferma dell'avvenuta ricezione della stessa e la conseguente notizia della fine dell'interrupt al microprocessore trasmittente. Questo è stato fatto mediante l'utilizzo della linea di selezione d'indirizzo del porto di Input che resetta il flip-flop di tipo D.

Supposto di utilizzare una tecnica I/0 Memory Mapped e di aver assegnato ai porti di I/0 lo stesso indirizzo (4000, ciò è possibile in quanto i sistemi sono distinti) e che sia il programma principale del trasmettitore che

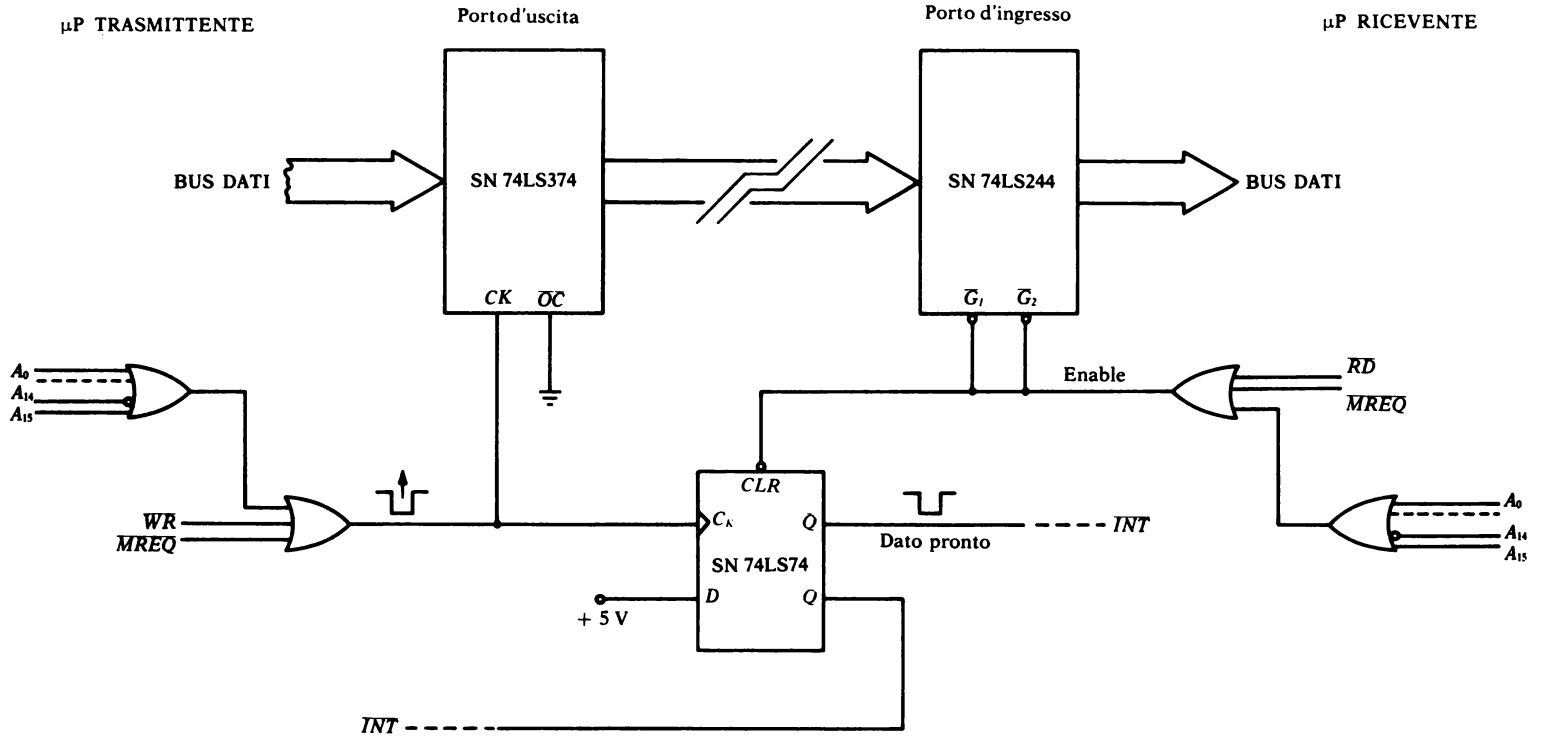


FIG. 66. - Circuito di interfaccia per la trasmissione dei dati tra due microcomputer.

quello del ricevitore siano stati abilitati all'interruzione mascherabile (per esempio nel modo 1) mediante le opportune istruzioni, la subroutine d'interrupt del microprocessore ricevente è quella di fig. 67.

```
PUSH AF          Salva i contenuti dei registri
PUSH HL

LD HL, (Punt)    Carica in HL l'indirizzo della locazione di memoria in cui deve essere immagazzinato il dato

LD A, (4000)     Abilita il porto di Input e disabilita la linea d'interrupt del microprocessore ricevente

LD (HL), A       Scrivi il dato nella locazione assegnata

INC HL

LD (Punt), HL    Aggiorna il puntatore

POP HL           Ripristina il contenuto dei
POP AF           registri


EI
RETI
```

FIG. 67. - Subroutine d'interrupt del ricevitore.

La subroutine del ricevitore viene eseguita non appena l'ingresso $\overline{INT}$ del microprocessore ricevente viene attivato (livello logico basso). L'esecuzione dell'istruzione di caricamento LD A, (4000), oltre ad abilitare il porto d'ingresso, resetta il flip-flop ed attiva il piedino $\overline{INT}$ del microprocessore trasmittente permettendo a quest'ultimo l'esecuzione della relativa subroutine d'interrupt (fig. 68). Durante l'esecuzione di tale subroutine viene abilitato il porto di uscita a trasferire l'informazione lungo la linea.

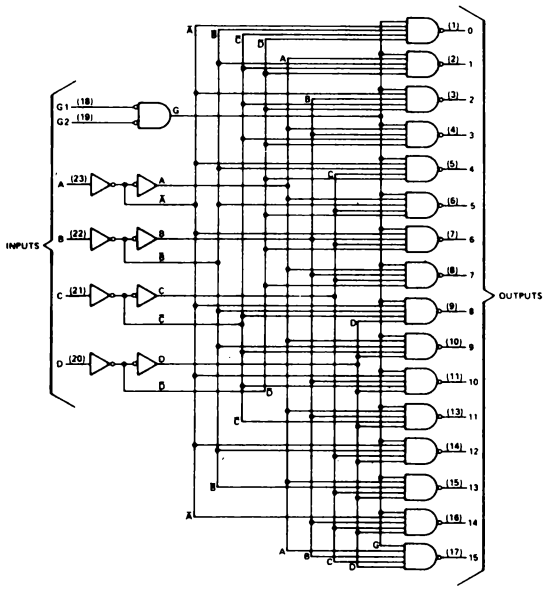
```
PUSH AF
PUSH HL
LD HL, (Punt)      Carica l'indirizzo del dato
                   da trasmettere

LD A, (HL)

LD (4000), A       Abilita il porto di uscita e
                   la linea d'interrupt del rice-
                   vitore per la trasmissione di
                   un nuovo dato (l'uscita del
                   flip-flop è rimessa a zero)

INC HL

LD (Punt), HL      Aggiorna il puntatore

POP HL

POP AF

EI

RETI
```

FIG. 68. - Subroutine d'interrupt del trasmettitore.

Per far si che la comunicazione fra i due sistemi abbia termine *occorre inserire via software un contatore di byte,* che azzerandosi non appena l'informazione voluta è stata interamente trasmessa, blocca la trasmissione. Naturalmente in sostituzione degli Or a 16 ingressi di fig. 66 si possono utilizzare semplicemente due Not, collegandoli per esempio alle linee A_{14} dei bus degli indirizzi, nel caso di sistemi con memoria riservata all'indirizzamento non superiore a 32 K (paragrafo 1).

12. Interfaccia per il collaudo di circuiti integrati

In questo paragrafo viene esaminato un dispositivo capace di collaudare il funzionamento delle porte logiche di circuiti integrati a 14 pin utilizzando la tecnica di gestione di I/0 Memory Mapped. Il circuito utilizzato è quello di fig. 70 in cui la selezione del porto d'ingresso e di quello d'uscita d'indirizzo rispettivamente 4001 e 4000, è stata effettuata mediante l'integrato SN 74154 (demultiplexer da 4 a 16 linee) della Texas.



FUNCTION TABLE

INPUTS						OUTPUTS														
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = high level, L = low level, X = irrelevant

FIG. 69. - Schema funzionale (a) e tabella della verità (b) dell'SN 74154 (Texas).

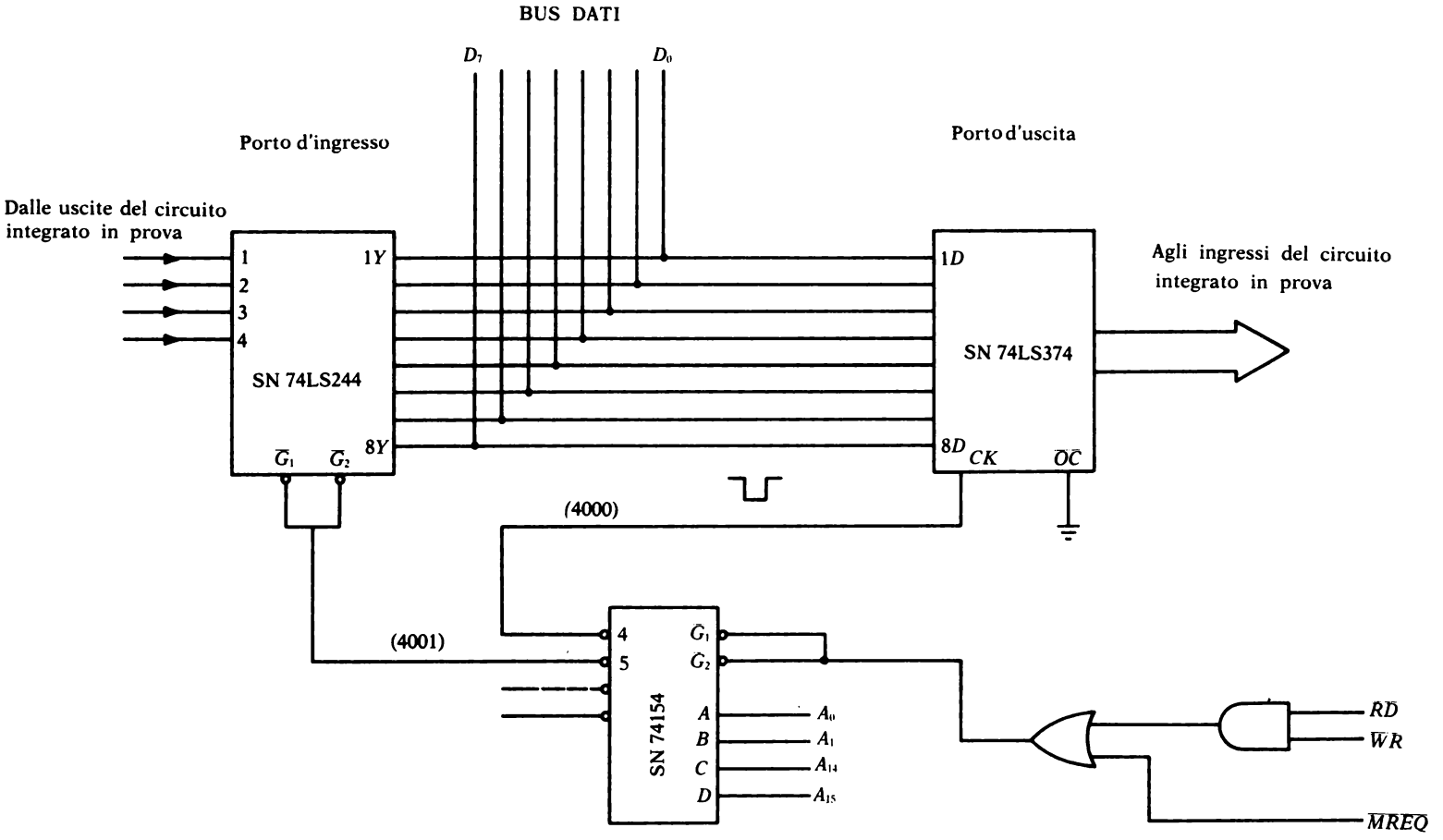


FIG. 70. - Circuito di collaudo per integrati a 14 pin ad otto ingressi e quattro uscite.

Dallo schema di fig. 70 si può notare che non appena viene eseguita l'istruzione LD (4000),A il demultiplexer viene abilitato ($\overline{WR}$ = $\overline{MREQ}$ = 0 e di conseguenza $\overline{G_1}$ = $\overline{G_2}$ = 0) ed è selezionata l'uscita 4 (in queste condizioni A_{15} = 0, A_{14} = 1, A_1 = A_0 = 0) che abilita il porto d'uscita. Allo stesso modo l'esecuzione dell'istruzione LD A, (4001) porta all'abilitazione del porto d'ingresso ($\overline{G_1}$ = $\overline{G_2}$ = 0, D = B = 0, C = A = 1).

L'uso del demultiplexer come decodificatore di indirizzo comporta una occupazione di una larga zona di memoria fittizia ma offre il vantaggio di una decodifica molto semplificata rispetto a quella necessaria per indirizzare univocamente i porti di I/0 (parag. 1).

Il programma di collaudo dei circuiti integrati si basa sulla memorizzazione delle tabelle della verità (metodo tabulare) e sul confronto dei risultati ottenuti dando tutte le possibili configurazioni all'ingresso del circuito integrato in prova. Il programma così come il flow-chart riportati in seguito sono relativi al collaudo di integrati con porte logiche And, Or, Xor, Nand e Nor a due ingressi. Le configurazioni da dare in ingresso a tali circuiti sono memorizzate in una tabella indicata con TAB IN mentre le configurazioni che ci si aspetta in uscita vengono scritte in una tabella indicata con TAB OUT. Il porto di uscita è interamente collegato con gli otto ingressi dell'integrato in prova mentre quello d'ingresso ha quattro pin liberi in quanto le uscite dell'integrato sono soltanto quattro. La TAB IN degli integrati sopra menzionati è uguale per tutti in quanto le combinazioni possibili ai due ingressi di una porta logica sono 00, 01, 10 ed 11. Ciò comporta che agli otto ingressi dell'integrato in prova debbono essere inviate le configurazioni.

00 00 00 00
01 01 01 01
10 10 10 10
11 11 11 11

che in esadecimale valgono rispettivamente 00, 55, AA ed FF

La tabella d'ingresso è quindi la seguente:

TAB IN
00
55
AA
FF

Le configurazioni delle tabelle d'uscita dipendono invece dalla funzione logica svolta dalla porta. Così per esempio, se l'integrato in prova è un SN 7408 (quadruplo And a due ingressi) in corrispondenza delle configurazioni d'ingresso 00,55, AA ed FF le configurazioni d'uscita sono rispettivamente uguali a 00,00,00 e 0F. Questo perché soltanto l'ultima configurazione d'ingresso 11 11 11 11 (si esegua il prodotto logico di ogni coppia d'ingressi dei quattro And costituenti l'integrato) dà come risultato la configurazione d'uscita 00 00 11 11.

In fig. 71 sono riportate le tabelle di uscita per alcuni integrati dove i quattro ingressi non utilizzati del porto d'ingresso sono stati assunti uguali a zero.

TAB OUT

SN 7408 (And)	SN 7432 (Or)	SN 7400 (Nand)	SN 7402 (Nor)	SN 7486 (Xor)
00	00	0F	0F	00
00	0F	0F	00	0F
00	0F	0F	00	0F
0F	0F	00	00	00

FIG. 71.

Nelle fig. 73 e 74 sono riportati rispettivamente il flow-chart ed il programma per il collaudo delle porte logiche appena considerate. Al termine dell'esecuzione del programma la locazione di memoria 0800 conterrà 00 se il collaudo dell'integrato ha avuto esito positivo oppure 01 in caso contrario.

Naturalmente l'esito del test può essere visualizzato su un display, per esempio visualizzando la lettera B se l'integrato testato è buono oppure la lettera E in caso contrario, mediante l'aggiunta del circuito di fig. 72 allo schema di fig. 70 utilizzando l'istruzione LD (4002), A al posto della istruzione LD (0800), A e caricando l'accumulatore invece che con 00 e 01 rispettivamente con 7C ed F9 sempre facendo riferimento al programma di fig. 74. La tecnica di I/0 è ancora quella di Memory Mapped ed il display è stato considerato come periferico con indirizzo di selezione pari a 4002.

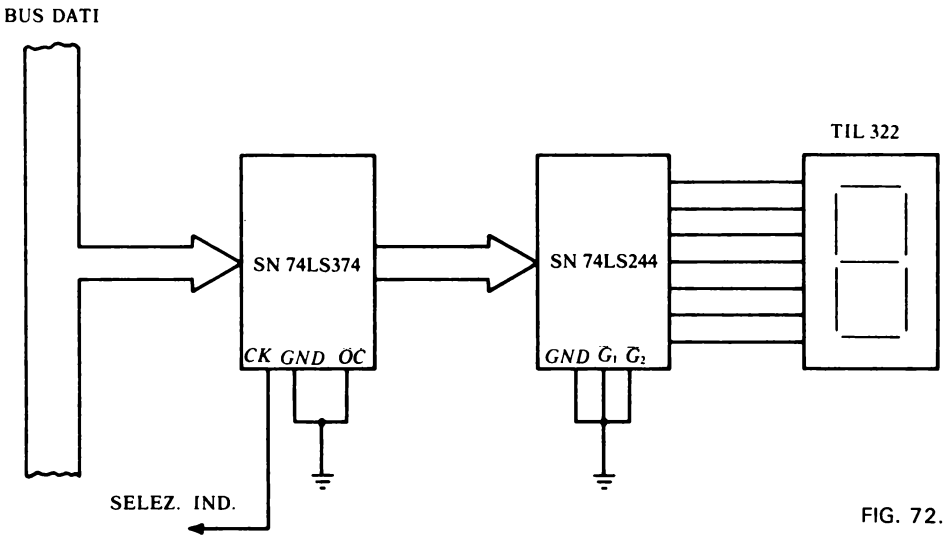


FIG. 72.

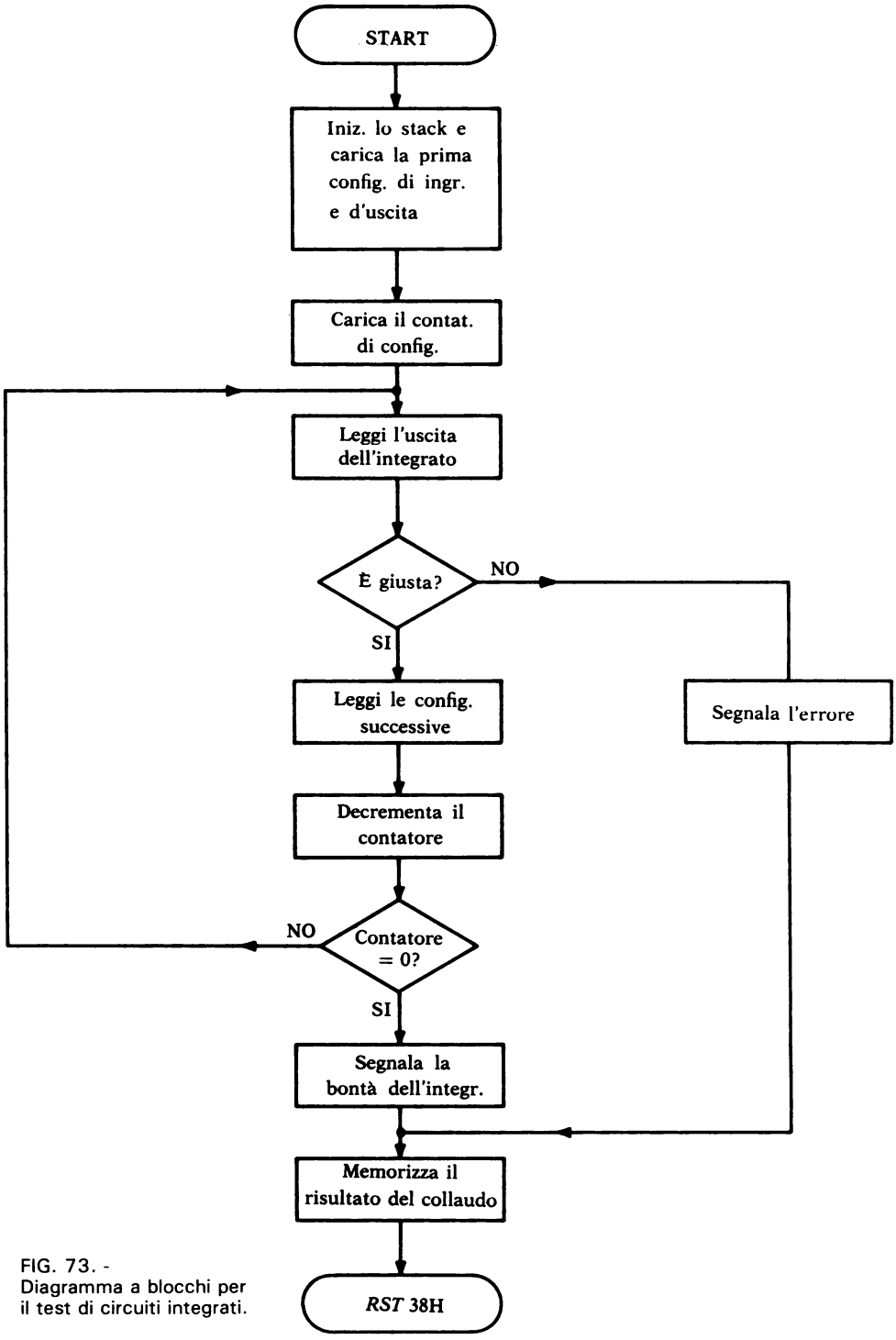


FIG. 73. - Diagramma a blocchi per il test di circuiti integrati.

Il funzionamento del circuito di fig. 72 è il seguente: non appena viene eseguita l'istruzione LD (4002), A il risultato del collaudo viene posto sul bus dei dati e trasferito tramite l'SN 74LS374, abilitato dall'uscita 6 del demultiplexer, al display.

L'integrato SN 74LS244 in questo caso viene utilizzato semplicemente come buffer necessario per pilotare il display che si presuppone a catodo comune.

Programma:

	LD SP, 0A00	Definisci l'area di stack
	LD DE, TAB IN	Carica l'indirizzo della tabella d'ingresso
	LD HL, TAB OUT	Carica l'indirizzo della tabella d'uscita
	LD B, 04	Carica il contatore di configurazioni
P_1	LD A, (DE)	
	LD (4000), A	Trasferisci il contenuto di A agli ingressi dell'integrato
	LD A, (4001)	Leggi le uscite dell'integrato
	AND 0F	Maschera A con 0F (la parte pesante di A diventa 0 mentre la leggera, che contiene le uscite, rimane inalterata)
	CP (HL)	Confronta la configurazione d'uscita con quella desiderata scritta in TAB OUT
	JP NZ, Error	Salta ad Error se non è uguale
	INC DE	
	INC HL	
	DJ NZ P_1	Decrementa il contatore e salta a P_1 se è diverso da zero
	LD A, 00	Memorizza 00 se il test è buono
P_2	LD (0800), A	Poni il risultato del collaudo in 0800
	RST 38 H	Ridai il controllo al sistema operativo del microcomputer (programma monitor)
Error	LD A, 01	Memorizza 01 se il testo non è buono
	JP P_2	Poni il risultato del collaudo in 0800

FIG. 74.

Il circuito di fig. 70 ha lo svantaggio di poter collaudare soltanto una piccola gamma di circuiti integrati essendo strettamente legato al numero degli ingressi e delle uscite dei porti di I/0 utilizzati. Così ad esempio, se l'integrato da collaudare è l'SN 7442 (decodificatore BCD-decimale) a 16 pin, occorrono due porti di ingresso ed uno di uscita in quanto possiede 10 uscite e quattro ingressi. Inoltre in questo caso la stesura del programma non può più essere fatta adottando il metodo tabulare (osservazione della

tabella della verità) ma deve essere fatta secondo il metodo funzionale, cioè riferendosi alla funzione svolta dall'integrato. In fig. 75 è infine riportato lo schema di un circuito di interfaccia a 16 linee di I/0 per il collaudo di integrati con elevato numero di ingressi ed uscite. Tale circuito può anche essere realizzato per mezzo di componenti LSI particolari quali lo Z 80 PIO nel caso di utilizzo della famiglia Z 80.

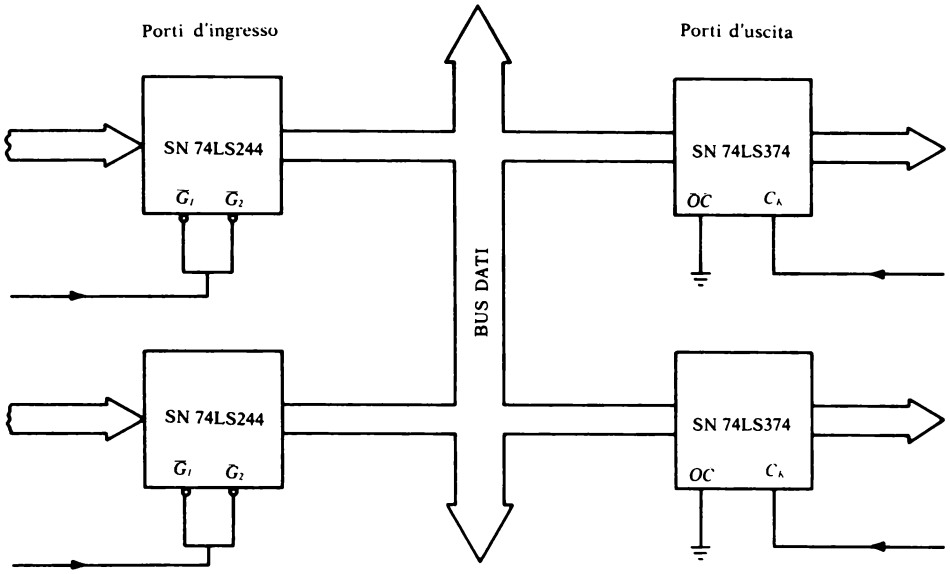


FIG. 75. - Circuito di interfaccia a 16 linee di I/O (la decodifica degli indirizzi è quella di fig. 70).

13. Interfaccia di una tastiera

Molto spesso occorre interfacciare una tastiera alfanumerica con un sistema a microprocessore, in fig. 77 è mostrato lo schema di un circuito capace di leggere i tasti di una tastiera, strutturata in una matrica 5×4 (20 tasti), e memorizzare il tasto premuto tramite programma. Il porto d'ingresso è letto mediante la tecnica di I/0 Memory Mapped. La selezione del porto d'ingresso e quella delle colonne avviene decodificando i bit del bus degli indirizzi. In particolare A_{15} ed A_{14} decodificano le colonne tramite il decoder SN 74LS42 (Texas) e tutti gli altri bit invece il porto d'ingresso (fig. 76).

Selez. Colonne				Selezione del porto d'ingresso														
Indiriz. Esadecim.	Colonna selezionata	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
2000	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6000	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8000	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
C000	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

FIG. 76. - Tabella di selezione del porto d'ingresso e delle colonne.

La chiusura di un tasto viene vista dal porto d'ingresso come uno 0 e l'apertura come 1. Il funzionamento del dispositivo è il seguente:
se ad un certo istante viene eseguita l'istruzione LD A, (2000) mentre il tasto 5 della prima colonna è premuto viene selezionata tale colonna ed abilitato l'SN 74LS244, l'informazione 11101111 presente ai suoi ingressi (l'ingresso 4D è a massa perché il tasto 5 è chiuso) viene caricata nell'accumulatore della CPU. Eseguendo poi via software una scansione dell'accumulatore è possibile individuare il tasto premuto della colonna selezionata.

Naturalmente tramite programma saranno poi sequenzialmente selezionate tutte le altre colonne e ripetute le stesse operazioni. In fig. 79 è mostrato un elementare flow-chart di gestione della tastiera in esame. La protezione contro i rimbalzi dei tasti anziché via hardware, può essere effettuata tramite software andando a rileggere più volte la riga su cui si è trovato il tasto premuto e verificando ogni volta di avere la stessa configurazione di bit dopo aver impostato il tempo massimo di rimbalzo previsto (di solito intorno ai 15 ms). Se più tasti sono stati premuti simultaneamente l'accumulatore conterrà più di uno zero, anche questa situazione, se non desiderata, può essere rimossa mediante programma. Se durante la scansione della tastiera nessun tasto è premuto l'accumulatore conterrà tutti 1 (FF) ed il contatore di colonne 0. In caso contrario il successivo programma, dopo aver identificato il tasto premuto servendosi del contenuto del contatore di colonne e della configurazione letta lungo la colonna, procede all'interpretazione del dato e alla esecuzione delle operazioni ad esso associate. Il programma di gestione della routine della tastiera è riportato in fig. 80.

Un inconveniente del circuito di fig. 77 si ha quando sono premuti più tasti nella stessa colonna o nella stessa riga. Infatti se per esempio sono premuti i tasti 4 e C della colonna 3 e la 3 viene interrogata, gli ingressi 3D e 5D del buffer tri-state sono messi al livello logico 0 come desiderato. Se poi viene interrogata la colonna 2 ed il tasto 3 è premuto (fig. 81), essendo il tempo di pressione del tasto molto maggiore del tempo di scansione del-

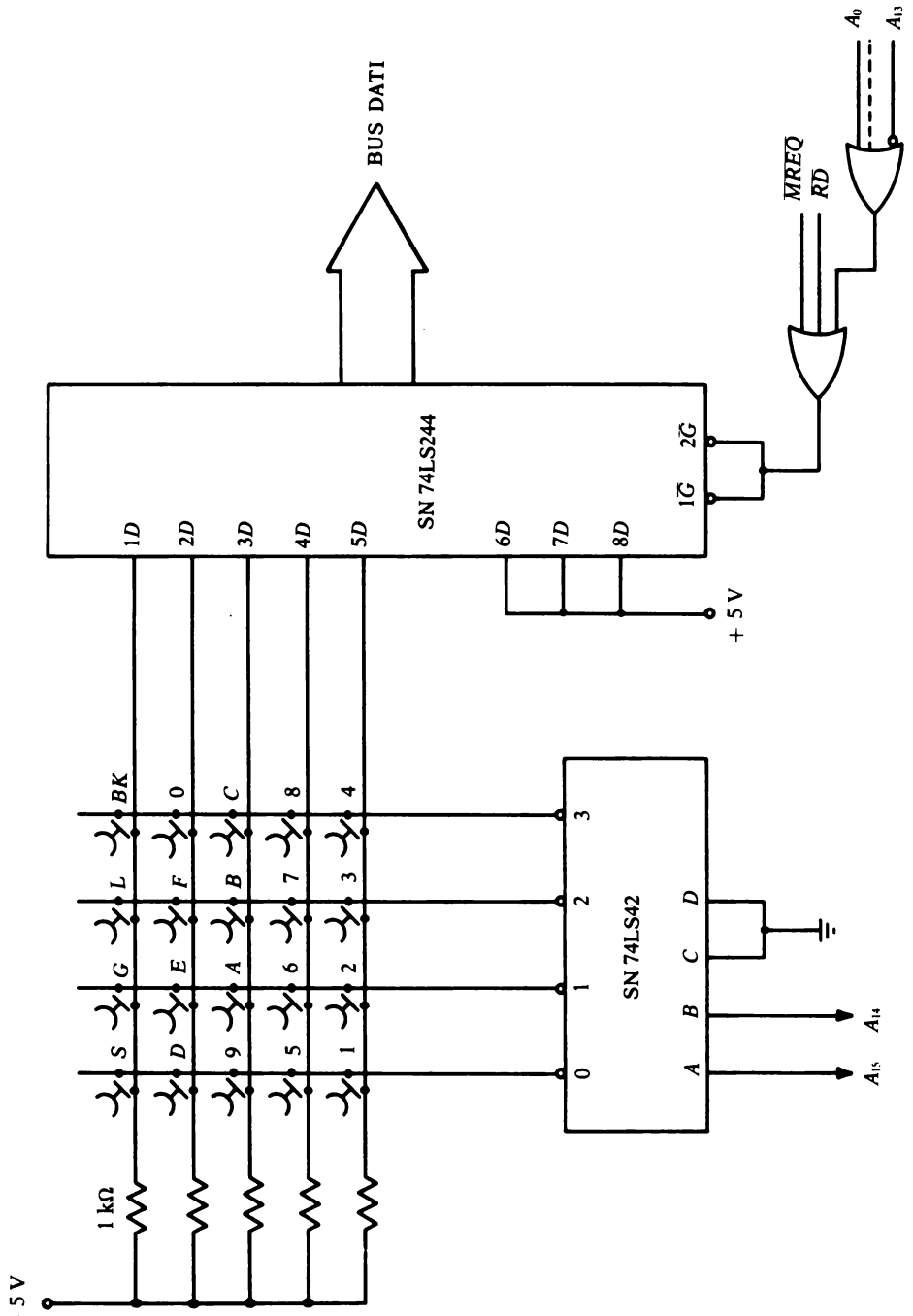
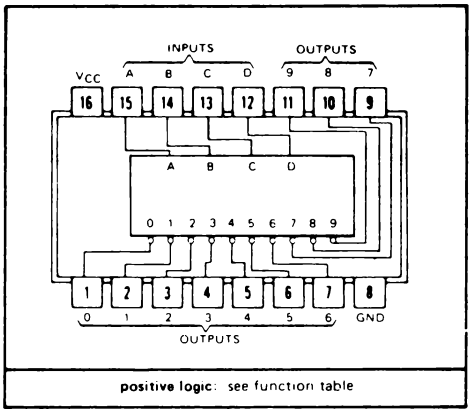


FIG. 77. - Circuito di interfaccia per una tastiera 5 × 4.



NO.	BCD INPUT				DECIMAL OUTPUT									
	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	L	L	L	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	H	H	H	H	H	H	H	L
INVALID	H	L	H	L	H	H	H	H	H	H	H	H	H	H
	H	L	H	H	H	H	H	H	H	H	H	H	H	H
	H	H	L	L	H	H	H	H	H	H	H	H	H	H
	H	H	L	H	H	H	H	H	H	H	H	H	H	H
	H	H	H	L	H	H	H	H	H	H	H	H	H	H
	H	H	H	H	H	H	H	H	H	H	H	H	H	H

FIG. 78. - Piedinatura (a) e tabella della verità (b) dell'SN 74LS42

la tastiera, gli ingressi 3D e 5D risultano ancora bassi (cortocircuito tra la V_{0H} e la V_{0L} con corrispondente danneggiamento della porta logica) come se anche il tasto B fosse premuto.

Un modo per eliminare questo inconveniente consiste nell'inserimento dei diodi al germano ($V_{AK} \cong 0{,}2$ volt in conduzione) ad ogni incrocio come in fig. 82. In queste condizioni infatti, come può essere facilmente verificato, i livelli logici agli ingressi 3D e 5D sono quelli corretti (alto e basso rispettivamente).

La scelta dei diodi al germanio è dovuta al fatto di dover mantenere il livello logico basso ad un valore riconoscibile come tale dall'integrato. Infatti se si fossero usati dei diodi al silicio la tensione $V_{0L} + V_{AK} \cong 0{,}8$ volt potrebbe non essere più riconosciuta come livello logico basso dall'ingresso del porto di input usato. Per poter utilizzare dei diodi al silicio occorre invertire i diodi ed effettuare la scansione della tastiera con degli 1 anziché che con degli 0. Le resistenze da 1 K Ω presenti nello schema hanno la solita funzione di limitare la corrente assorbita I_{0L} al di sotto di quella massima consentita al decodificatore.

È da tenere presente che esistono in commercio numerosi circuiti LSI che permettono l'interfacciamento diretto di tastiere e display con sistemi a microprocessore, l'8279 della INTEL è uno di questi. In fig. 83 è mostrato uno schema di applicazione in cui tale integrato interfaccia una tastiera, strutturata in matrice 8×8, ed un display. Per la descrizione del funzionamento di tale integrato e dello schema di fig. 83 si rimanda il lettore alla consultazione del manuale specifico della casa costruttrice.

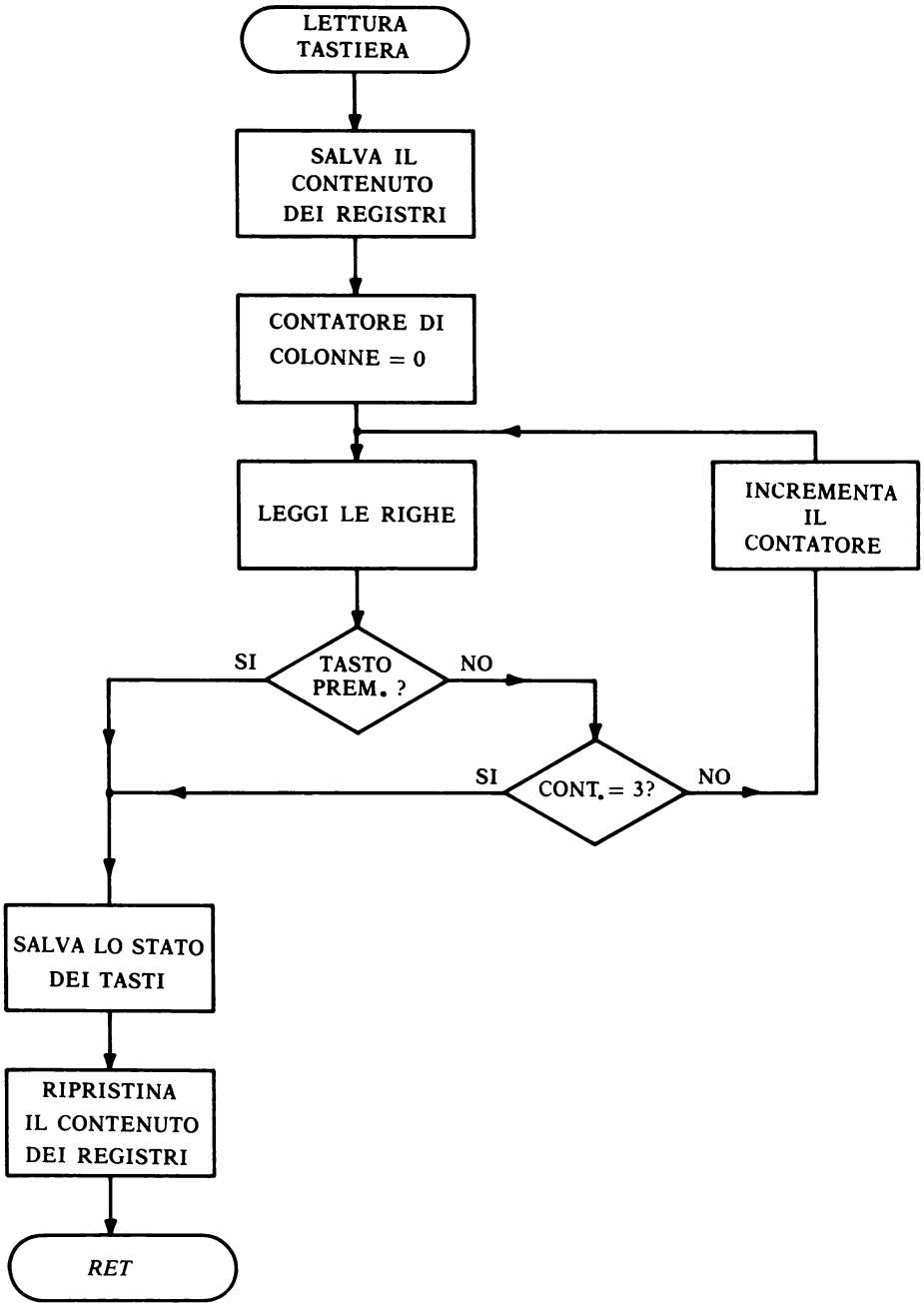


FIG. 79. - Routine di gestione di una tastiera.

Programma:

```
        PUSH AF          Salva il contenuto dei registri
        PUSH BC
        PUSH DE
        PUSH HL
        LD B, 00         Azzera il contatore di colonne
        LD A, (2000)     Seleziona la prima colonna (colonna zero)
        CP FF            Controlla se vi sono tasti premuti
        JP NZ P1         Salta a P1 se vi sono tasti premuti
        INC B
        LD A, (6000)     Seleziona la seconda colonna (colonna uno)
        CP FF
        JP NZ P1
        INC B
        LD A, (8000)     Seleziona la terza colonna (colonna due)
        CP FF
        JP NZ P1
        INC B
        LD A, (C000)     Seleziona l'ultima colonna (colonna tre)
P1      LD (nn), A       Memorizza e salva lo stato dei tasti nella lo-
                         cazione di memoria puntata da nn
        POP HL
        POP DE
        POP BC
        POP AF
        RET
```

FIG. 80.

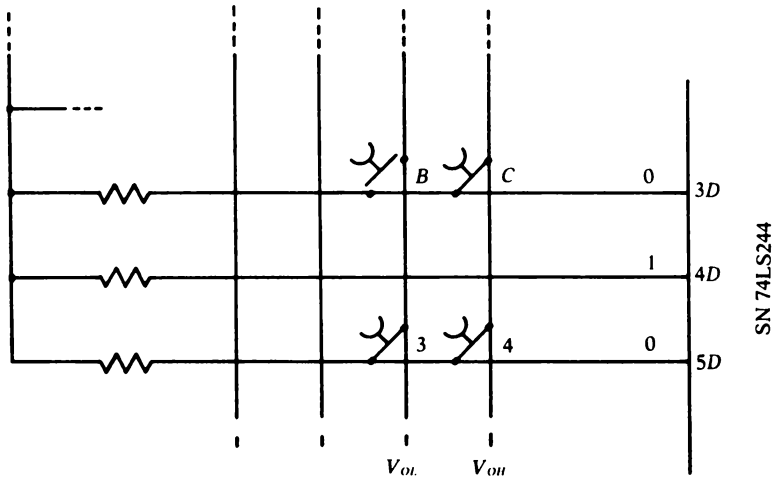


FIG. 81. Selezione della colonna 2 mentre i tasti della 3 sono ancora premuti (livelli logici non corretti).

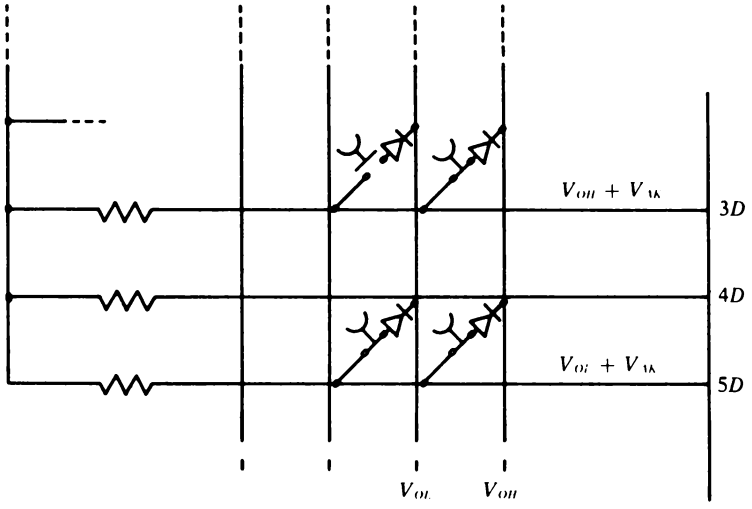
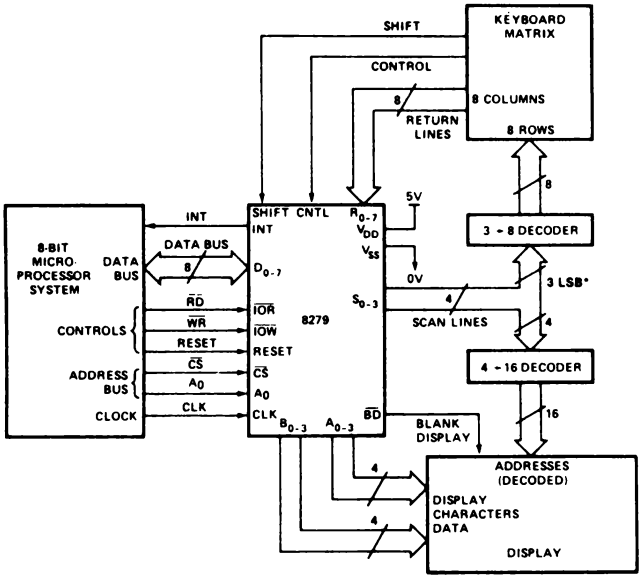


FIG. 82. - Selezione della colonna 2 mentre i tasti della 3 sono ancora premuti (livelli logici corretti).



*Do not drive the keyboard decoder with the MSB of the scan lines.

FIG. 83. - Applicazione dell'8279 (INTEL).

14. CTC Z 80 (Counter Timer Circuit)

La serie Z 80 della SGS comprende una vasta gamma di componenti di supporto per microcomputer necessari per realizzare sistemi a più alte prestazioni e a basso costo senza dover ricorrere, o quanto meno ricorrendo in limiti accettabili, ad una logica esterna.

Lo Z 80 CTC, dispositivo contatempi-contaeventi programmabile a quattro canali indipendenti, svolge le funzioni di contatore e temporizzatore per sistemi a microprocessore basati sullo Z 80.

Tale componente se usato come contatore (counter mode) è in grado di accettare degli impulsi esterni e di interrompere il programma principale dopo un certo numero di impulsi, stabilito dall'operatore, se la CPU è stata predisposta a gestire l'interruzione in modo 2. Se invece il CTC viene utilizzato come temporizzatore (timer mode) è capace di contare gli impulsi di clock del sistema e di inviare un segnale d'interrupt dopo un certo numero di impulsi di clock.

Le principali caratteristiche dello CTC Z 80 sono le seguenti:

— Ogni canale può essere selezionato per operare sia in counter mode oppure in timer mode
— Un registro costante di tempo ricarica il Down-Counter (contatore di decremento) ogni volta che quest'ultimo viene azzerato.
— La lettura del Down-Counter effettuata dalla CPU indica il numero di conteggi che restano per arrivare a zero.
— Per il funzionamento in timer mode può essere triggerato sia positivamente che negativamente.
— Tre canali possiedono uscite (zero count/time out Φ) in grado di pilotare transistori Darlington.
— Sono possibili interruzioni programmabili in entrambi i modi di funzionamento.
— È provvisto di una logica interna di interruzione di tipo daisy chain.
— Tutti gli ingressi e le uscite sono completamente compatibili con dispositivi TTL.

In fig. 84 è mostrato lo schema a blocchi dello Z 80 CTC. La struttura interna consiste in una interfaccia per la CPU, una logica interna di controllo, quattro canali per il conteggio e una logica di controllo interruzione.

Ogni canale ha la possibilità di generare un vettore d'interruzione per saltare automaticamente alla relativa routine d'interrupt quando ciò viene richiesto. La priorità d'interruzione è determinata dal numero del canale, con 0 numero più prioritario.

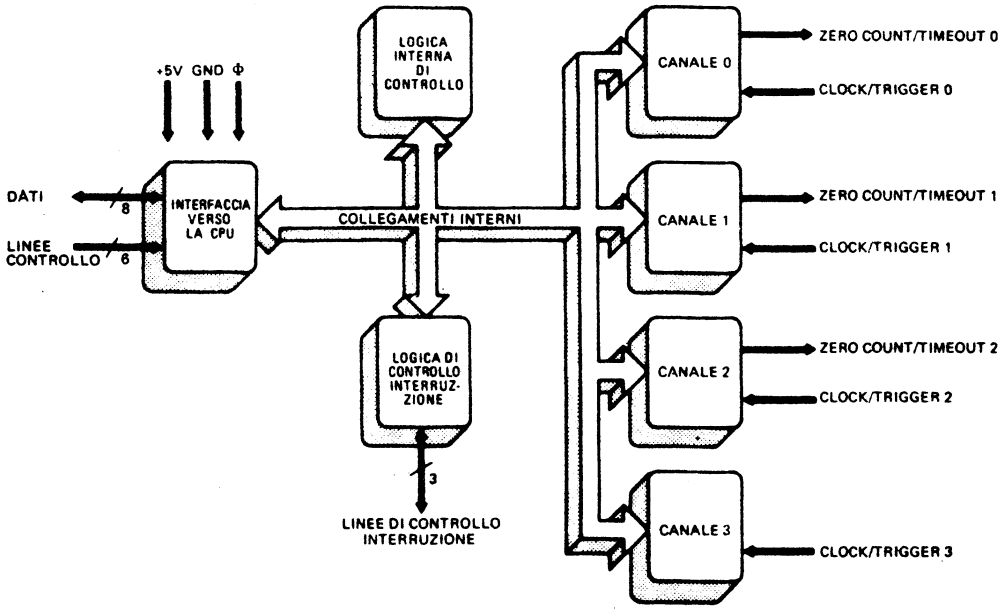


FIG. 84. - Schema a blocchi del CTC Z 80 (dal manuale tecnico SGS).

La logica del canale è costituita da due registri, due contatori e da una logica di controllo come mostrato in fig. 85.

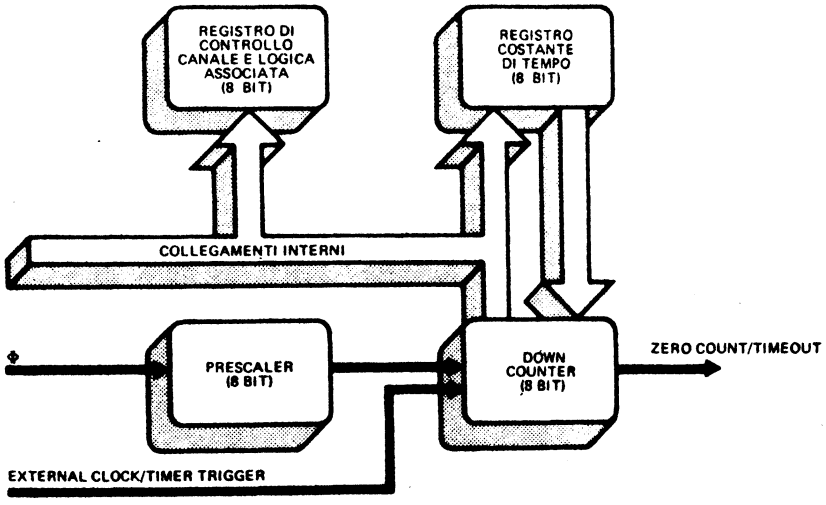


FIG. 85. - Schema a blocchi di un canale.

I registri sono un registro costante di tempo (time costant register) ed un registro di controllo di canale (channel control register) entrambi ad otto bit. I contatori sono un Down Counter, leggibile dalla CPU, ed un Prescaler (riduttore di conteggio) anch'essi ad otto bit. Le funzioni da loro svolte sono le seguenti:

Registro costante di tempo

Questo registro, programmabile dalla CPU, inizializza e ricarica il Down Counter ogni volta che quest'ultimo si azzera.

Registro controllo canale

Il registro controllo di canale, programmabile dalla CPU, seleziona il modo e le condizioni di operazione del canale.

Registro contatore di decremento

Il Down Counter è caricato dal registro costante di tempo all'atto della inizializzazione ed ogni volta che scende a zero. In qualsiasi momento la CPU può leggere il numero di conteggio rimasto per arrivare a zero.

Prescaler

È un registro, programmabile dalla CPU, capace di dividere la frequenza del sistema per 16 o per 256 per decrementare il Down Counter. Il Prescaler viene usato soltanto nel funzionamento del timer mode.

Per quanto riguarda la programmazione, le caratteristiche tecniche e le applicazioni specifiche del CTC Z 80 si rimanda il lettore alla consultazione del manuale tecnico edito dalla casa costruttrice.

15. Fan-Out e Fan-In nei sistemi a microprocessore

I concetti di fan-in e fan-out relativi al collegamento di porte logiche usuali (Vol. Elettronica Digitale cap. 2°) possono essere estesi ai dispositivi a microprocessore dove vengono scambiati segnali di uscita tra i pin dei vari componenti.

Normalmente questi segnali debbono essere bufferizzati in quanto, per esempio, facendo riferimento a sistemi Z 80 i componenti di tale famiglia possono pilotare uno o due al massimo carichi TTL. Questo comporterebbe una forte limitazione nell'hardware del sistema, limitazione che però viene eliminata bufferizzando tutte le linee dei vari bus.

Di solito tutti i microcomputer, dal più semplice al più sofisticato, hanno tutte le linee dei bus bufferizzate all'atto della costruzione.

Per avvertire di ciò l'operatore normalmente tali linee sono precedute

nella loro identificazione dalla lettera B (buffer).

Così per esempio BA_0 e $\overline{BIORQ}$ stanno ad indicare che la linea del bus d'indirizzi A_0 e l'uscita $\overline{IORQ}$ della CPU sono entrambe bufferizzate.

Naturalmente all'interno del software del microcomputer, dove la logica è rappresentata da una sequenza di istruzioni, non ha senso parlare di fan-in e fan-out in quanto lo stato di un digit binario può essere il risultato di un numero più o meno grande di operazioni logiche cosicché in questo caso il fan-in ed il fan-out risultano entrambi infiniti.

ESERCIZI PROPOSTI

1. Progettare un circuito di decodifica d'indirizzo, per una gestione I/O memory mapped che occupi due K di memoria fittizia a partire dalla locazione 0600.

2. Progettare un circuito di decodifica d'indirizzo, per una gestione I/O isolato per la locazione di memoria 0100.

3. Programmare la porta B del PIO Z 80 in modo bidirezionale.

4. Determinare il numero di PIO indirizzabili dalla CPU Z 80.

5. Tenendo presente la seguente parola di controllo delle interruzioni si ricavi la parola di maschera che abilita le linee A_7, A_5 ed A_3 della porta A del PIO Z 80 specificando come viene generata una richiesta d'interruzione.

1	1	1	1	0	1	1	1

6. Progettare un circuito capace di collegare in configurazione daisy chain più di quattro PIO Z 80.

7. Mediante l'impiego di due integrati 8212 (INTEL) si realizzi un bus driver bidirezionale.

8. Si trasformi un bus bidirezionale in due unidirezionali.

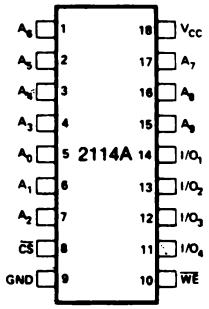
9. Progettare un circuito che resetti contemporaneamente PIO e Z 80 CPU senza dover togliere la tensione di alimentazione.

10. Ricavare la TAB IN e la TAB OUT per l'integrato SN 7404.

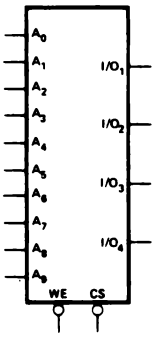
11. Codificare il programma di fig. 73 supponendo di caricarlo a partire dalla locazione di memoria 0100. Gli indirizzi di TAB IN e TAB OUT siano rispettivamente uguali a 0300 e 0400.

12. Da due RAM 2114 (1024 x 4 bit, INTEL fig. 86) ricavarne una da 1 K X 8 compatibile con il bus dati del microprocessore Z 80.

PIN CONFIGURATION



LOGIC SYMBOL



PIN NAMES

A_0–A_9	ADDRESS INPUTS	V_{CC} POWER (+5V)
$\overline{WE}$	WRITE ENABLE	GND GROUND
$\overline{CS}$	CHIP SELECT	
I/O_1–I/O_4	DATA INPUT/OUTPUT	

BLOCK DIAGRAM

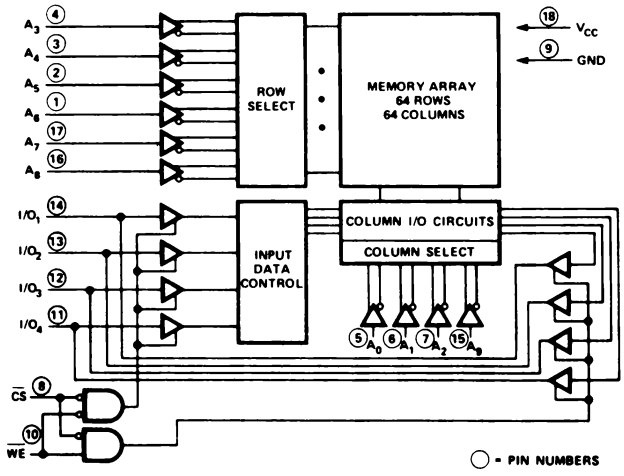


FIG. 86. - RAM 2114 (dal manuale tecnico INTEL).

CAPITOLO QUINTO

LA TRASMISSIONE E IL RUMORE

1. Generalità

Nel capitolo precedente sono stati esaminati circuiti in cui l'informazione veniva trasferita da un dispositivo all'altro, un carattere alla volta, inviando contemporaneamente tutti i bit del carattere su tanti canali quanti erano i bit trattati.

Questo tipo di trasmissione, detto parallelo, viene però usato assai raramente, per via delle numerose linee necessarie, e solo nei sistemi di un certo costo nei quali è richiesta una elevata velocità di esecuzione. In genere

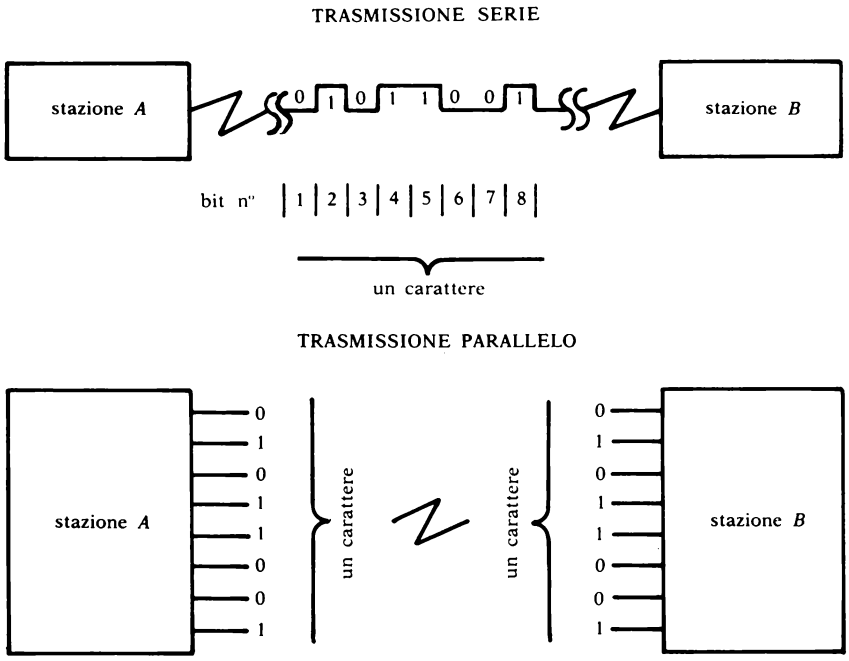


FIG. 1. - Trasmissione in serie e parallelo.

nella trasmissione dell'informazione tra microcomputer e periferici e tra microcomputer stessi, specialmente se la loro distanza è medio alta, si preferisce usare un sistema di trasmissione seriale in quanto in questo caso si ottiene una notevole semplificazione di tutto l'equipaggio di trasmissione e del canale di comunicazione.

Nella trasmissione seriale, infatti, i bit del carattere vengono trasmessi uno alla volta per ciascun intervallo di clock.

2. Trasmissione asincrona

Quando l'informazione digitale viene trasmessa attraverso un canale di comunicazione, i segnali elettrici che ne sono portatori debbono avere incorporati degli elementi che ne permettano la sincronizzazione al sistema generale, altrimenti non potrebbero essere decodificati.

La comunicazione asincrona utilizza come metodo di sincronizzazione delle informazioni addizionali formate generalmente da un bit di start (0 logico) e da uno o più bit di stop (1 logico).

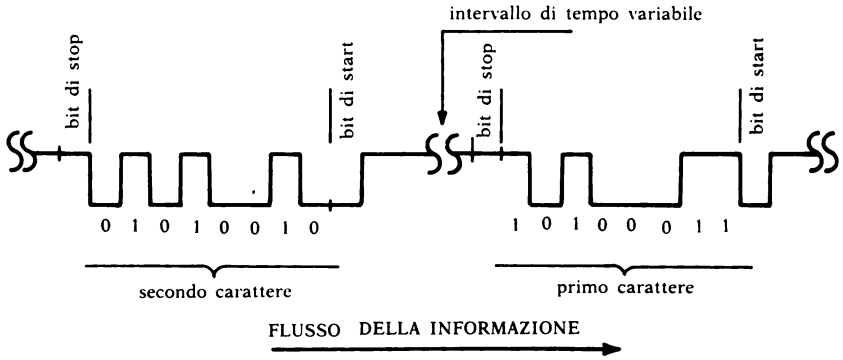


FIG. 2. - Trasmissione asincrona in serie.

Questo tipo di trasmissione viene utilizzato per colloquiare con terminali di bassa o media velocità in quanto ben si adatta al ritmo con cui l'informazione è generata, per esempio l'ingresso di dati tramite una tastiera. Infatti in questo caso si trasmette un carattere alla volta, introducendo i bit di start e stop per segnalare l'istante d'inizio e di termine della trasmissione del carattere.

In fig. 2 è mostrato un esempio di caratteri trasmessi in modo seriale asincrono. La linea di trasmissione in condizioni di riposo è al livello logico alto, l'inizio di un carattere trasmesso è indicato da una transizione della linea dal livello logico alto al livello logico basso (bit di start). Subito dopo sono inviati gli n bit costituenti il carattere (con n che va normal-

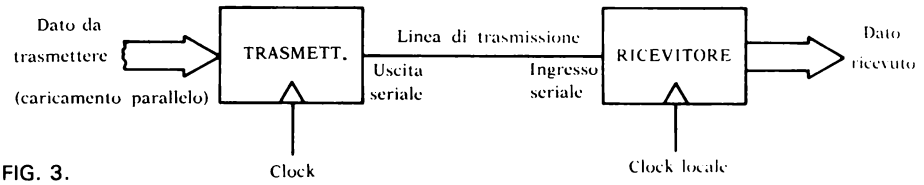


FIG. 3.

mente da 5 ad 8 esclusi i bit di start e di stop) seguiti da uno o più bit di stop. Il ricevitore (fig. 3), che conosce il numero di bit costituenti il carattere, al sopraggiungere del bit di start fa partire un clock locale di frequenza uguale a quello del trasmettitore e riconosciuto il bit di stop riporta al livello logico alto la linea di trasmissione.

Per non perdere l'informazione la frequenza del clock locale non deve discostarsi di un certo ammontare rispetto a quella del trasmettitore, ammontare che dipende dal tipo di sincronizzazione adottato.

Esempio

Si supponga di aver trasmesso il codice 10101010, il periodo di clock del trasmettitore vale:

$$T_T = \frac{1}{f_T}$$

e la durata della trasmissione

$$10T_T$$

dove il numero dieci è comprensivo dei bit di start e di stop.

Se il clock del ricevitore parte dopo un certo tempo (ad esempio mezzo periodo) dal momento di arrivo del bit di start l'errore massimo che si può commettere per non perdere l'informazione è di mezzo bit su un totale di otto. In questo caso nel tempo $10\ T_T$ il clock del ricevitore compie 9 periodi e mezzo di quello del trasmettitore, la frequenza f_R del ricevitore non deve quindi scendere al di sotto del 95% di quella del trasmettitore:

$$f_R = \frac{9,5}{10} \quad f_T$$

Nel caso di sincronizzazione ogni 100 byte la precisione della frequenza del ricevitore dovrà essere molto maggiore valendo in questo caso:

$$f_R = \frac{999,5}{1000} \quad f_T$$

In fig. 4 è mostrato lo schema a blocchi di un semplice dispositivo capace di effettuare la trasmissione asincrona seriale tra due sistemi per caratteri da sei bit partendo da un caricamento parallelo dell'informazione tramite uno shift-register ed un flip-flop. Normalmente l'uscita non negata dello shift è al livello logico 1 e alto l'ingresso di inibizione del clock ($\overline{EC}$), in queste condizioni la linea seriale è messa al livello logico alto tramite l'uscita $\overline{Q}$ che agisce sul preset (attivo basso) del flip-flop.

La trasmissione del carattere lungo la linea avviene poi abilitando il clock (il primo impulso di clock carica l'informazione nello shift register, i successivi fanno scorrere ogni volta i bit immagazzinati verso destra).

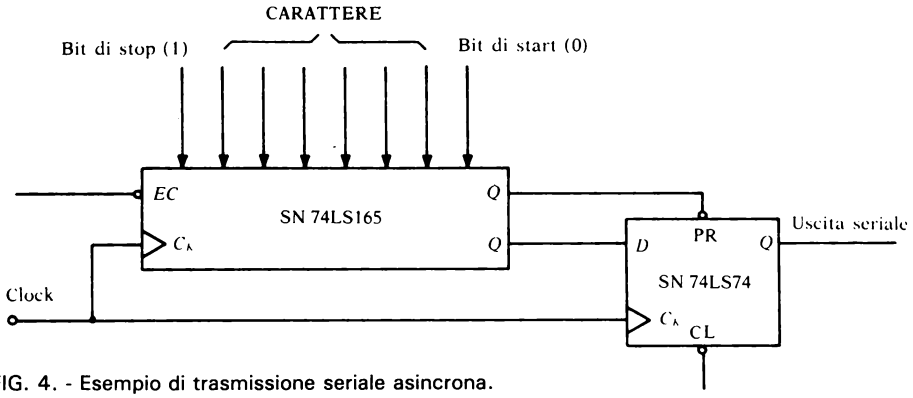


FIG. 4. - Esempio di trasmissione seriale asincrona.

3. Trasmissione sincrona

Nel sistema di comunicazione sincrono l'informazione viene trasmessa in modo continuo ed uniforme in base ad un clock di riferimento, per cui non è più necessario l'uso dei bit di start e di stop. In questo sistema si trasmette generalmente un gruppo di più caratteri e, molto spesso, all'interno del treno di bit vengono introdotti particolari segnali per indicare ad esempio la lunghezza del testo, il controllo degli errori etc.

Attualmente i sistemi di comunicazione sincroni vengono usati quando è richiesta una elevata velocità. In fig. 5 è mostrato un treno di bit trasmesso attraverso un canale con funzionamento sincrono.

Nella trasmissione sincrona si hanno due diverse possibilità di sincronizzazione; la prima consiste nell'aggiungere una linea esterna che si porta al livello logico alto in corrispondenza del primo dato (external synchronism), ed in questo caso è necessaria una linea in più; oppure si può trasmettere un carattere (o due caratteri se il sistema è bisincrono) di sincronismo che naturalmente non deve poi comparire nel testo del messaggio.

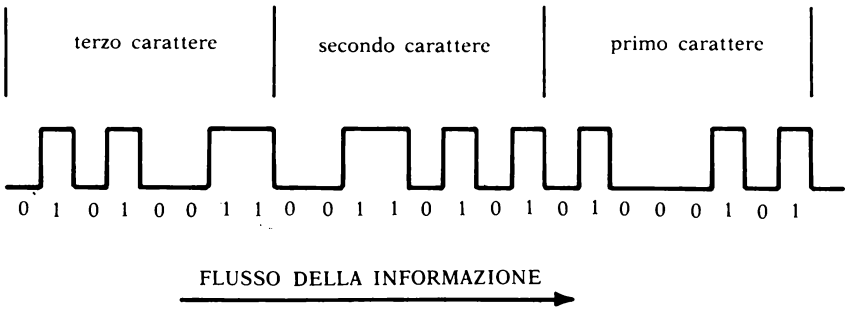


FIG. 5. - Trasmissione sincrona seriale.

In quest'ultimo caso il ricevitore legge la linea e se il carattere letto è quello di sincronismo dà via libera alla ricezione. In condizioni di riposo il trasmettitore invia continuamente lungo la linea, caratteri di sincronismo, la trasmissione inizia quando tali caratteri vengono a mancare.

Protocolli sincroni

La comunicazione dei dati richiede l'uso di una grammatica o *protocollo* per assicurare la corretta sequenza ed integrità del messaggio da trasmettere. Il protocollo per la trasmissione asincrona è composto dai bit di start e di stop mentre ci sono vari differenti protocolli di uso comune per quella sincrona.

Quelli comunemente usati sono:

— protocollo BSC o BISYNC (Binary Synchronous Communication)
— protocollo DDCMP (Digital Data Communication Message Protocol)
— protocollo HDLC (High-level Data Link Controls)
— protocollo ADCCP (Advanced Data Communication Control Procedures)
— protocollo SDLC (Synchronous Data Link Control).

I protocolli sincroni sono composti da caratteri di controllo e dal testo. I caratteri di controllo sono speciali sequenze di bit usate per portare informazioni circa il processo stesso di comunicazione. Per esempio, il DDCMP usa il carattere di sincronismo SYN per sincronizzare il ricevitore con il trasmettitore. Nel codice ASCII, SYN ha la configurazione 0010110.

L'SDLC ha solamente un carattere di controllo, chiamato carattere di flag, che ha la configurazione fissa: 01111110.

Il testo, invece, è costituito dai dati che devono essere trasmessi. La sola funzione di un protocollo è di trasmettere il testo.

Molto semplicemente si può paragonare un protocollo per la trasmissione dei dati, alla grammatica usata per scrivere un libro.

Funzione dei protocolli sincroni

Nella tecnica di trasmissione sincrona, il trasmettitore ed il ricevitore sono sincronizzati mediante l'uso di uno speciale carattere chiamato SYN. Non appena il ricevitore individua questo carattere, lo usa per mettere in fase il controllo con i successivi dati inviati.

FORMATTAZIONE: determinate posizioni o *campi* in ciascun blocco di messaggi, sono riservate per specifici tipi di informazione. Per esempio, ogni blocco può avere una testata seguita dal testo stesso ed infine da una terminazione, come è mostrato nella figura 6.

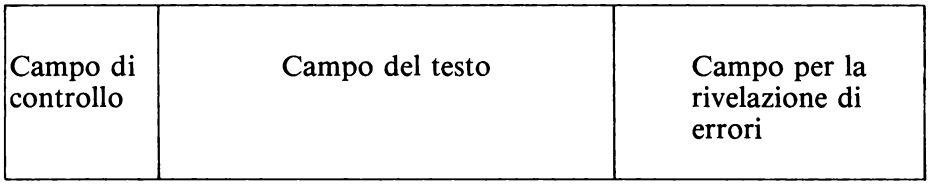


FIG. 6. - Un tipico formato di protocollo sincrono.

La testata contiene l'indirizzamento, la sequenza del blocco e informazioni sul flag le quali assicurano che i dati arrivino al giusto ricevitore, nell'appropriata sequenza e siano interpretati correttamente.

La terminazione generalmente contiene informazioni circa la rilevazione degli errori, che vengono utilizzate dal ricevitore.

UTILIZZAZIONE DELLA LINEA: il protocollo spesso determina la massima efficienza con la quale può essere usata la linea di trasmissione, vale a dire il numero di dati che possono essere trasmessi in un dato periodo di tempo. Per esempio, il protocollo BISYNC non può trarre vantaggio da un collegamento full-duplex poiché il ricevitore può dare solo risposte di riconoscimento. Per contro, il DDCMP è realizzato in modo che ciascuna apparecchiatura possa spedire e ricevere dati allo stesso tempo.

HANDSHAKING: le tecniche di comunicazione sincrona implicano un costante dialogo tra gli apparecchi di comunicazione per cui i termini *trasmettitore* e *ricevitore* possono adattarsi di volta in volta ad entrambe le apparecchiature. Tale dialogo, che mantiene la sincronizzazione e permette al dispositivo ricevente di inviare messaggi circa l'esattezza o meno del blocco di dati ricevuti, è spesso chiamato *handshaking* (letteralmente: stretta di mano).

Alcuni di tali dialoghi sono realizzati dai protocolli, mentre altri dai modem. Per esempio, quando un dispositivo è pronto a trasmettere (ready to send), dapprima invia un carattere di richiesta di trasmissione, il ricevitore, allora, trasmette un carattere di permesso di trasmissione (clear to send). A questo punto il trasmettitore invia i caratteri di SYN per stabilire

la sincronizzazione. Dopo che ciascun blocco è stato spedito, il ricevitore cerca gli eventuali errori ed invia un segnale di riconoscimento positivo o negativo.

RIVELAZIONE E CORREZIONE DEGLI ERRORI: il protocollo deve inoltre assicurare che il testo ricevuto sia completo e senza errori.

Ciò viene effettuato incorporando in esso delle procedure coinvolgenti un algoritmo applicato a ciascun blocco o testo, per generare un campo per la rivelazione degli errori. Nel prossimo paragrafo sono più ampliamente trattate queste procedure. Da notare che la trasmissione sincrona è una tecnica orientata al blocco, per cui se avviene un errore di trasmissione, deve essere ritrasmesso l'intero blocco.

TRASPARENZA DEI DATI: i protocolli sincroni usano speciali configurazioni di bit per indicare caratteri di controllo, ma ci sono molti modi per rappresentare i dati ed alcuni codici usano queste stesse configurazioni per dati che non sono caratteri di controllo. Ogni protocollo ha così, un modo di trasmettere il testo che possa permettere al ricevitore di guardare la configurazione di bit come un carattere di controllo.

4. La rilevazione degli errori

Per controllare l'esattezza del segnale ricevuto possono essere fatti dei test particolari mediante l'utilizzo di componenti LSI specifici. I controlli più diffusi sono il *controllo di parità orizzontale,* il *controllo di parità verticale* ed il *controllo ridondante circolare.*

Controllo di parità orizzontale

In questo tipo di controllo si aggiunge un bit agli n che compongono l'informazione. Il bit aggiunto è 0 od 1 a seconda che si abbia un numero pari o dispari di 1. Il ricevitore vede la parità e si confronta con il bit di parità trasmesso, se questo confronto è negativo viene segnalato l'errore.

Controllo di parità verticale

Questo controllo consiste nell'inserire in coda al messaggio un byte che analizza la parità verticale.

Si supponga, per esempio, di trasmettere un messaggio costituito da tre dati da otto bit:

	1	0	0	0	0	0	0	0	1	
parità pari	1	0	1	1	1	0	1	0	1	messaggio
→	0	0	0	1	1	1	0	1	0	

parità → sull'ultimo byte	0	0	1	0	0	1	1	1	0	byte di controllo di parità verticale

Se il messaggio ricevuto è il seguente:

1	0	0	0	0	0	0	0	1
1	0	**0**	1	1	0	1	0	1
0	0	0	1	1	1	0	1	0
0	0	1	0	0	1	1	1	0

Il ricevitore calcola la parità orizzontale ottenendo 100 e si accorge che il secondo byte è errato. A questo punto passa poi a controllare la parità verticale e ottenuto come risultato 00001110 individua il bit errato nel secondo byte. Mediante questo controllo si potrebbero evidenziare anche due bit errati all'interno dello stesso byte senza però conoscere quali sono. Se il ricevitore individua un errore in orizzontale e nessuno in verticale ciò vuol dire che vi è un errore nel byte di parità verticale.

Controllo ciclico ridondante o CRC (Cyclic Redundancy Checking)

Questo tipo di test è basato sulla divisione, eseguita in modo seriale, del messaggio trasmesso per un dato polinomio.

Il controllo CRC può essere eseguito soltanto nel caso di trasmissione sincrona.

Per la costruzione del codice ciclico, inizialmente si genera un polinomio di grado n - 1 se sono n i bit da trasmettere, che viene chiamato f (x) con x variabile che viene utilizzata per la sua costruzione. Successivamente si divide tale polinomio per un altro g(x) detto polinomio generatore di grado inferiore, ottenendo un quoziente q (x) ed un resto r (x):

$$\frac{f(x)}{g(x)} = q(x) + r(x)$$

Il polinomio r (x), che è il CRC, viene trasmesso subito dopo f (x) formando un pacchetto unico.

Se lungo la linea di trasmissione non sono stati commessi errori, il ricevitore, dividendo il polinomio ricevuto per g (x), deve ottenere resto zero.

In questo modo, gli unici errori che non vengono rivelati sono quelli, estremamente rari, che danno origine ad una serie di bit che è divisibile, senza resto, per g (x).

Esistono in commercio dei chip *LSI* che implementano diversi tipi di *CRC* e che riescono a controllare la correttezza dei dati ricevuti in modo da facilitare il lavoro di progetto.

5. Modo di trasmissione dell'informazione.

La forma che la trasmissione della informazione realizza all'interno del canale di comunicazione può avere diverse configurazioni indicate come: simplex, semi-duplex, duplex-completo o eco-plex.

Trasmissione in modo simplex

In questo caso si può solamente trasmettere in un senso unico, ossia solo trasmettere o solo ricevere ma non ambedue.

Nel teleprocesso questo modo di trasmissione è raramente usato poiché non permette alcun tipo di interazione. La sua utilizzazione si limita a quei sistemi nei quali interessa solamente inviare informazioni che non necessitano di risposta.

Trasmissione in modo semi-duplex

In questo caso il canale è capace di trasferire l'informazione in ambo i sensi (ricevere e trasmettere) però non simultaneamente. È un metodo di trasmissione molto comune che però ha lo svantaggio di introdurre ritardi

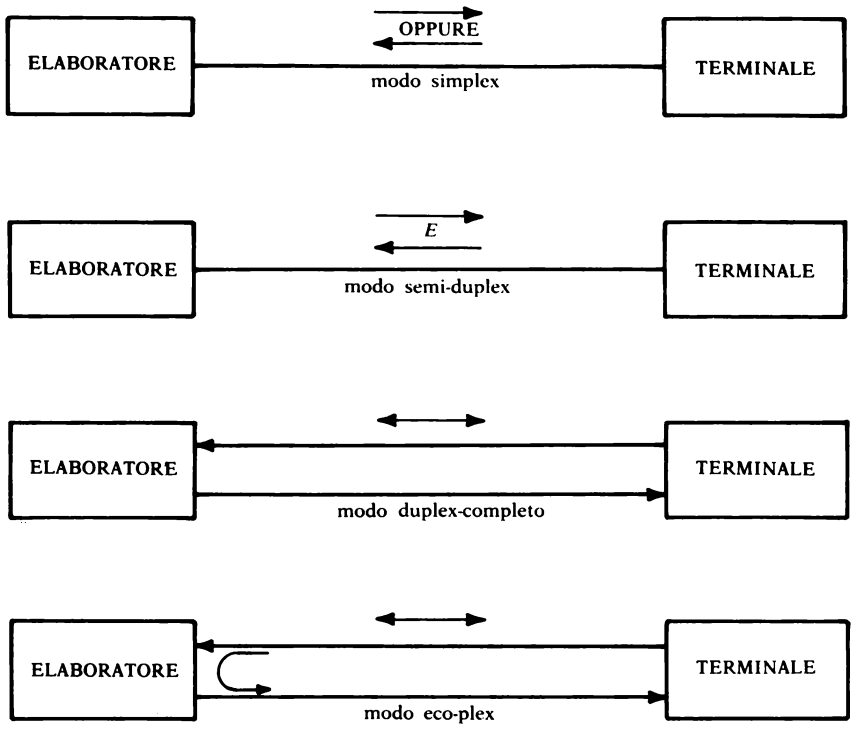


FIG. 7. - Metodi di trasmissione nel canale di comunicazione.

addizionali quando occorre commutare dalla trasmissione alla ricezione e viceversa.

Trasmissione in modo duplex-completo.

In un canale che funziona in modo duplex-completo si può sostenere la comunicazione in entrambi i sensi simultaneamente. Si può dire che è equivalente a due canali funzionanti in modo semi-duplex, uno per trasmettere e l'altro per ricevere, eliminando così la necessità di commutazione. La maggior parte dei sistemi di teleprocesso di bassa e media velocità funzionano in questo modo, specialmente se il supporto è il canale telefonico pubblico.

Trasmissione in modo eco-plex

Questa è una variante del modo duplex-completo, nella quale tutta l'informazione che è trasmessa dal dispositivo remoto, è immediatamente ritrasmessa indietro dall'elaboratore, per verificare l'esattezza della trasmissione, da cui il nome «eco». Il modo eco-plex dà innegabili vantaggi all'utente ma provoca una maggiore saturazione del canale di trasmissione usato.

Nella fig. 7 sono illustrati i modi di trasmissione descritti precedentemente.

6. Velocità di trasmissione

La velocità di operazione di un sistema di teleprocesso dipende tanto dalla velocità di operazione del computer (da questa dipende il tempo di risposta), quanto dalla velocità di operazione dell'equipaggio di trasmissione.

La velocità di quest'ultimo viene misurata in bit/secondo ma a volte la capacità di un canale viene indicata in *BAUD* che indica la massima velocità con la quale un segnale elettrico può cambiare stato all'interno di un canale. Per questa ragione i termini bit/sec e *BAUD* sono a volte confusi e usati indistintamente, però l'unico caso in cui l'equivalenza è vera è quando una variazione del segnale elettrico corrisponde ad un bit di informazione. La tendenza attuale è di costruire sistemi più efficienti che assegnano più bit a ciascuno stato del segnale portante l'informazione, per esempio un *dibit* (due bit per stato), un *tribit*, un *tetrabit*, ecc. In questi casi un bit per secondo non equivale più ad un *BAUD* ma quest'ultimo equivale a 2 bit/sec, 3 bit/sec e così via rispettivamente.

Si definiscono sistemi a bassa velocità (molto comuni per il loro basso costo), quelli che rimangono al di sotto dei 1200 bit/sec. Le velocità più generalmente usate sono quelle di 110,150,300 (la usuale) e 600 bit/sec, per il modo duplex-completo.

I sistemi considerati di media velocità lavorano a 1200 o 1400 bit/sec. In

questa categoria si incontra la trasmissione asincrona in modo duplex-completo (1200 bit/sec), quando si usa come canale di comunicazione il canale telefonico.

Infine, i sistemi ad alta velocità comunemente sono del tipo sincrono e semi duplex e lavorano a 9600 bit/sec nel canale telefonico. Benintenso esistono dei sistemi che superano ampiamente tali velocità, come quelle usate per le intercomunicazioni dei computer (vari *megabit* al secondo), che però non sono tipiche della rete di teleprocesso e non possono funzionare nel canale telefonico.

7. I codici

Quando si vogliono trasmettere delle informazioni in forma digitale, siano esse composte da numeri, lettere o simboli particolari, un certo numero di bit (da un minimo di quattro ad un massimo di otto) può essere raggruppato in varie configurazioni che danno origine ai vari tipi di codice.

Quelli più semplici, composti da quattro bit, possono formare $2^4 = 16$ diverse configurazioni che a loro volta possono essere associate a simboli diversi.

Tra gli innumerevoli codici a quattro bit, utilizzati soprattutto nelle macchine da calcolo, i più noti sono mostrati nella tabella seguente.

Decimale	*BCD* 8 4 2 1	*Aiken* 2 4 2 1	*Gray*	*Eccesso 3*
0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 0 0 1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 0 1 0	0 0 1 1	0 1 0 1
3	0 0 1 1	0 0 1 1	0 0 1 0	0 1 1 0
4	0 1 0 0	0 1 0 0	0 1 1 0	0 1 1 1
5	0 1 0 1	1 0 1 1	0 1 1 1	1 0 0 0
6	0 1 1 0	1 1 0 0	0 1 0 1	1 0 0 1
7	0 1 1 1	1 1 0 1	0 1 0 0	1 0 1 0
8	1 0 0 0	1 1 1 0	1 1 0 0	1 0 1 1
9	1 0 0 1	1 1 1 1	1 1 0 1	1 1 0 0

Codice BCD (Binary Coded Decimal)

I quattro bit che lo compongono hanno peso 8, 4, 2, 1 come nel sistema binario puro, ed ogni cifra decimale di un qualsiasi numero viene rappre-

sentata dalla loro combinazione. Ad esempio, il numero 249 viene codificato in:

$$249 = 0010\ 0100\ 1001$$

Esiste anche una versione più complessa di questo codice che utilizza sette bit, corrispondenti a 128 diverse combinazioni, più un bit per il controllo di parità dispari.

Codice Aiken

È del tutto simile al *BCD*, ma i quattro bit hanno peso 2, 4, 2, 1 per cui il precedente numero 249 preso come esempio, viene codificato in:

$$249 = 0010\ 0100\ 1111$$

Ha il vantaggio di essere *autocomplementante,* ciò significa che complementando i singoli bit si ottiene automaticamente il complemento a nove, il che semplifica le operazioni di tipo aritmetico.

Per esempio il codice 1011 che equivale al numero 5, se complementato dà 0100 equivalente a 4.

Codice Gray

In questo codice, nel passaggio da una cifra a quella successiva, le combinazioni differiscono per un solo bit; è a volte utilizzato nei convertitori A/D.

Codice eccesso 3

È ottenuto sommando ad ogni cifra decimale il numero 3. Viene utilizzato poiché è anch'esso di tipo autocomplementante e perché facilita il controllo degli errori di trasmissione non avendo cifre con tutti zero nella lista.

Tra i codici a cinque bit i più noti sono quello Johnson in cui, nel passaggio da una cifra a quella vicina si ha un aumento o una diminuzione del numero di uno nella lista, ed il codice Baudot (*CCITT* n. 2) che viene utilizzato soprattutto nei sistemi di tipo telegrafico.

Nel teleprocesso, i codici comunemente usati sono due, il codice *ASCII (American Standard Code for Information Interchange)* ed il codice *CCITT* n. 5 *(Comitato Consultivo Internazionale per la Telegrafia e Telefonia)*. Il primo è lo standard Nord-Americano, mentre il secondo è quello internazionale.

Le seguenti tabelle mostrano le composizioni dei due codici. Come si vede sono entrambi ad otto bit (l'ottavo bit si usa per la rivelazione degli errori) e praticamente uguali, differenziandosi solamente per alcuni simboli e codici speciali di controllo.

Codice Baudot

		1	2	3	4	5
A	—	x	x			
B	?	x			x	x
C	:		x	x	x	
D	Chi è	x			x	
E	3	x				
F	A disposiz.	x		x	x	
G	A disposiz.		x		x	x
H	A disposiz.			x		x
J	Fine		x	x		
I	8	x	x		x	
K	(	x	x	x	x	
L	)		x			x
M	.			x	x	x
N	,			x	x	
O	9				x	x
P	0		x	x		x
Q	1	x	x	x		x
R	4		x		x	
S	'	x		x		
T	5					x
U	7	x	x	x		
V	=		x	x	x	x
W	2	x	x			x
X	/	x		x	x	x
Y	6	x		x		x
Z	+	x				x
Ritorno carrello					x	
Interlinea			x			
Cambio lettere		x	x	x	x	x
Cambio cifre		x	x		x	x
Spaziatura				x		
Non utilizzato						

Codice ASCII

b_4 ↓	b_3 ↓	b_2 ↓	b_1 ↓	COLUMN → / ROW ↓	0	1	2	3	4	5	6	7
				b_7 →	0	0	0	0	1	1	1	1
				b_6 →	0	0	1	1	0	0	1	1
				b_5 →	0	1	0	1	0	1	0	1
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	¦
1	1	0	1	13	CR	GS	–	=	M	]	m	}
1	1	1	0	14	SO	RS	·	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Codice CCITT N. 5

b_4	b_3	b_2	b_1		b_7 0	b_6 0	b_5 0	0	b_7 0	b_6 0	b_5 1	1	b_7 0	b_6 1	b_5 0	2	b_7 0	b_6 1	b_5 1	3	b_7 1	b_6 0	b_5 0	4	b_7 1	b_6 0	b_5 1	5	b_7 1	b_6 1	b_5 0	6	b_7 1	b_6 1	b_5 1	7
0	0	0	0	0	(1)				(2)				Spazio (27)				(0)				(4) (5)				P				a (4)				p			
0	0	0	1	1	SOH (26)				DC1 (6)				!				1				A				Q				a				q			
0	0	1	0	2	STX (29)				DC2 (6)				" (5)				2				B				R				b				r			
0	0	1	1	3	ETX (19)				DC3 (6)				# (34)				3				C				S				c				s			
0	1	0	0	4	EOT (17)				Arresto (28)				× (35)				4				D				T				d				t			
0	1	0	1	5	WRU (33)				NACK (23)				%				5				E				U				e				u			
0	1	1	0	6	ACK (10)				SCNC (30)				&				6				F				V				f				v			
0	1	1	1	7	Campanello (11)				ETB (18)				(5)				7				G				W				g				w			
1	0	0	0	8	BS (12)				CNCL (13)				(				8				H				X				h				x			
1	0	0	1	9	HT (21)				EM (16)				)				9				I				Y				i				y			
1	0	1	0	10	LF (22)				(2)				*				:				J				Z				j				z			
1	0	1	1	11	VT (32)				(2)				+				;				K				[(4)				k				(3)			
1	1	0	0	12	FF (20)				FS (32)				,				<				L				~ (4) (5)				l				(3)			
1	1	0	1	13	CR (14)				GS (31)				—				=				M				] (4)				m				(3)			
1	1	1	0	14	SO (25)				RS (31)				.				>				N				^ (4) (5)				n				(3)			
1	1	1	1	15	SI (24)				US (31)				/				?				O				— (36)				o				Obliterazione (15)			

8. USART (Universal Synchronous Asynchronous Receiver Transmitter)

Per l'interfacciamento seriale nei microcomputer vengono utilizzati di solito componenti specifici LSI quali l'UART (Universal Asynchronous Receiver Transmitter) o più frequentemente l'USART in quanto quest'ultimo può operare, a differenza del primo a cui non è permesso, anche in modo asincrono.

L'USART è in grado di convertire in forma seriale per la trasmissione caratteri prelevati dalla CPU in parallelo, viceversa può ricevere dei dati in forma seriale e convertirli in parallelo (fig. 8).

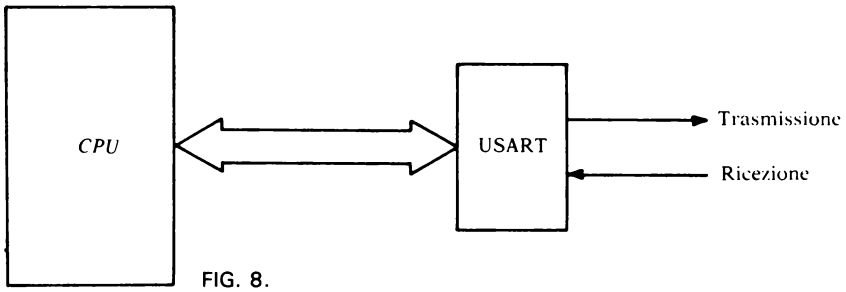


FIG. 8.

Tutto questo può essere fatto contemporaneamente all'esecuzione del programma principale da parte del microcomputer, tranne che durante la gestione della lettura del carattere ricevuto o il trasferimento del prossimo carattere da trasmettere in cui il programma principale viene interrotto dall'USART.

L'USART è quindi usato come dispositivo periferico indirizzabile tramite il bus del sistema ed è *programmabile* dalla CPU nel suo modo di operare (sincrono/asincrono, numero di bit per carattere, sincronizzazio-

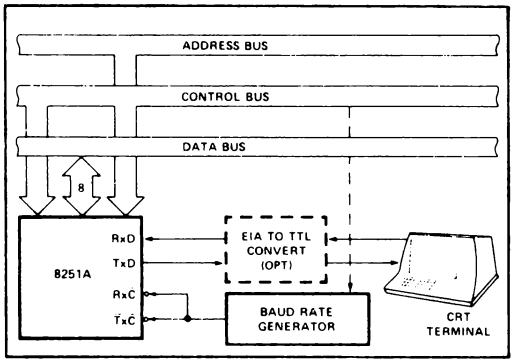


FIG. 9. - Interfaccia seriale asincrona per un terminale CRT (dal manuale tecnico INTEL).

ne esterna / interna, velocità di trasmissione / ricezione, numero di bit di stop etc.).

La CPU può leggere lo stato completo dell'USART in qualsiasi momento e controllare, per esempio, leggendo in un apposito registro se vi sono stati o meno errori di ricezione.

Uno degli USART a più larga diffusione e compatibile con lo Z 80 CPU è l'8251 della INTEL, nelle fig. 9, 10 e 11 sono mostrate alcune sue tipiche applicazioni.

Per quanto riguarda la descrizione, il modo di funzionamento e la programmazione di questo componente LSI a tecnologia NMOS si rimanda il lettore alla consultazione del manuale specifico edito dalla stessa casa costruttrice.

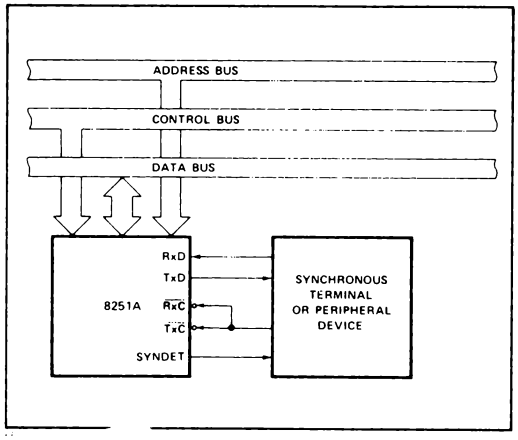


FIG. 10. - Interfaccia sincrona per un terminale o dispositivo periferico (dal manuale tecnico INTEL)

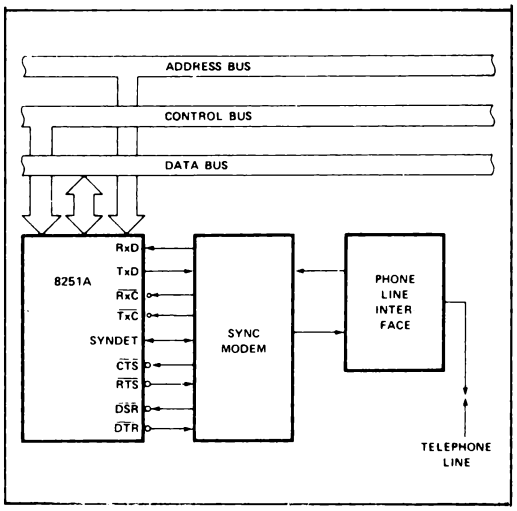


FIG. 11. - Interfaccia sincrona per una linea telefonica (dal manuale tecnico INTEL)

9. Il Modem

Il *modem* è il dispositivo elettronico che si utilizza per effettuare il trasferimento dell'informazione tra località distanti fra di loro. Il canale di comunicazione per il quale esso è realizzato è generalmente il canale telefonico come già indicato, e la sua funzione è di modificare l'informazione digitale in modo che possa essere trasmessa attraverso un canale analogico.

Il *modem* realizza questo processo per mezzo della modulazione ed in effetti il suo nome deriva dalla combinazione delle parole Modulatore-Demodulatore. Poiché deve essere in grado di stabilire delle comunicazioni con altri *modem* remoti, ciò avviene per mezzo dei *protocolli di comunicazione* che sono un'insieme di regole e specifiche che permettono di stabilire adeguatamente l'allaccio. Un allaccio è stabilito una volta che i *modem* sono interconnessi tramite il canale di comunicazione ed avviene la trasmissione dell'informazione digitale attraverso la modulazione di una portante analogica (di forma sinusoidale) che è facilmente trasmittibile.

La modulazione può essere di tre tipi: ampiezza, frequenza e fase. Altre funzioni che possono essere realizzate dal *modem* sono di mantenere la supervisione, ad ogni istante, dello stato dell'allaccio, trasformare l'informazione da asincrona a sincrona, controllare gli errori, ecc.

10. Interfacce digitali standard.

Con il nome di interfacce standard vengono individuate le regole funzionali e le configurazioni elettriche e meccaniche normalizzate, mediante le quali si realizza l'interconnessione fisica tra un dispositivo terminale chiamato *DTE (Data Terminal Equipment)* ed un dispositivo trasmittente chiamato *DCE (Data Communication Equipment)* che è generalmente il *Modem* per trasmissioni a lunghe distanze.

Tali interfacce adottano due diversi tipi di trasmissione chiamati: trasmissione in *banda base* e trasmissione a *modulazione di portante.*

Nella prima, i bit sono associati a variazioni dei valori di tensione o corrente in un circuito a corrente continua, mentre si parla di modulazione di portante per quei circuiti in corrente alternata dove i bit sono associati a variazioni di ampiezza, frequenza e fase, come ad esempio avviene per i *modem.*

In questo paragrafo verranno illustrate le più diffuse interfacce quali l'EIA n. RS-232-C (*Electrical Industries Association)* ed il 20 mA Current Loop, ma è necessario ricordare che oltre a questi si sono aggiunti i più recenti *EIA* RS-422-A e 423-A per tensioni bilanciate e sbilanciate, l'*IEEE* 488/GPIB *(Institute of Electrical and Electronic Engineers)* per strumenti di misura e molti altri anche non ufficialmente approvati. Fra questi ultimi vanno menzionati i vari bus per microprocessori.

PARAMETRI DELLE SPECIFICHE EIA			
CARATTERISTICHE	EIA RS—232—C	EIA RS—423	EIA RS—422
Forma d'impiego	single-ended	single-ended	differential
Lunghezza massima del cavo	50 ft	2.000 ft	4.000 ft
Data rate massimo	20 Kilobauds	300 Kilobauds	10 megabauds
Tensione massima d'uscita del driver, circuito aperto	± 25 V	± 6 V	6 V between outputs
Tensione minima d'uscita del driver, uscita caricata	± 5 to ± 15 V	± 3,6 V	2 V between outputs
Resistenza minima d'uscita del driver, spento	R_O = 300 Ohm	100 µA between —6 to + 6 V	100 µA between + 6 and —0 25 V
Uscita massima del driver, corrente di cortocircuito I/sc	± 500 mA	± 150 mA	± 150 mA
Slew rate d'uscita del driver	30 V/µsec max	Lo slew-rate dovrebbe essere basato sulla lunghezza del cavo e sul tasso di modulazione	no control necessary
Resistenza d'ingresso del receiver R_{IN}	3 to 7 KOhm	> 4 KOhm	> 4 KOhm
Soglie d'ingresso massimo del receiver	—3 to + 3 V	—0 2 to + 0 2 V	—0.2 to + 0.2 V
Tensione d'ingresso massimo del receiver	—25 to + 25 V	—12 to + 12 V	—12 to + 12 V

EIA RS-232-C.

Tutti i segnali che intervengono in questa interfaccia digitale per la connessione degli equipaggi coinvolti in un sistema di teleprocesso, sono mostrati nella tabella 1.

Fisicamente, l'interfaccia è costituita da un connettore a 25 piedini (*pin*) ai quali vengono assegnate le varie funzioni specifiche per lo standard Nord-Americano *EIA* RS-232-C e per la versione equivalente *CCITT,* raccomandazione V 24.

Ciascuna funzione può essere raggruppata in una delle quattro diverse categorie seguenti:

— circuiti di massa,

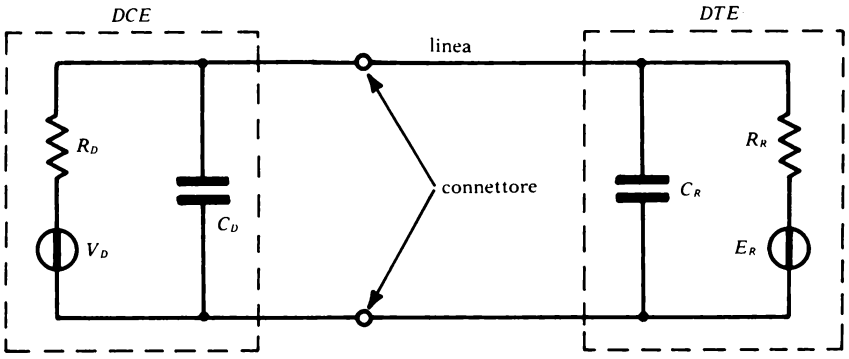


FIG. 12. - Circuito equivalente per l'interfaccia EIA RS-232-C.

— circuiti di dati,
— circuiti di controllo,
— circuiti di temporizzazione.

Infatti ciascun piedino, insieme al ritorno comune, costituisce un intero circuito il cui schema equivalente è mostrato nella fig. 12.

In essa, V_D è la tensione del circuito di pilotaggio in assenza del carico, R_D la sua resistenza interna e C_D la capacità parassita calcolata fino al punto fisico di interfaccia costituito dal connettore a 25 pin.

Per il circuito ricevente, E_R è la tensione residua a circuito aperto, R_R la resistenza di carico e C_R la capacità parassita comprendente anche quella del cavo fino al connettore.

Nella fig. 13 sono mostrati i livelli di tensione dei segnali specificati da questa interfaccia.

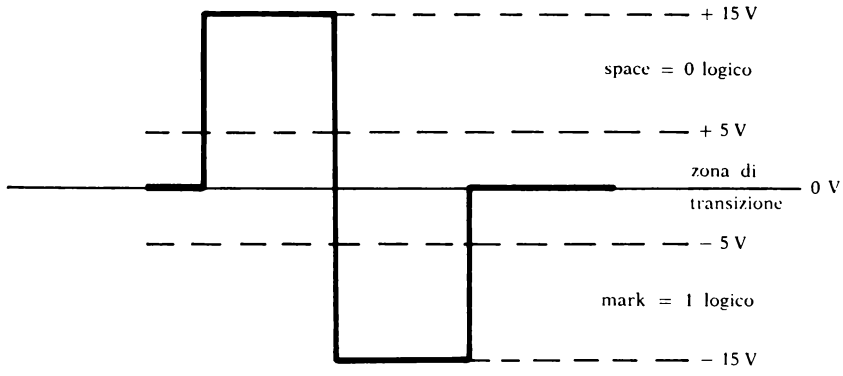


FIG. 13. - Livelli di tensione per lo standard RS-232-C.

L'intera escursione può essere suddivisa in tre diverse regioni corrispondenti a tre diverse condizioni di funzionamento, nel modo seguente:

regione compresa tra +5V e + 15 V = zero logico = ON = *SPACE*

regione compresa tra — 5V e — 15 V = uno logico = OFF = *MARK*

regione compresa tra — 5 V e + 5 V = zona di transizione.

Valori minimi e massimi delle caratteristiche elettriche del circuito di fig. 12 sono i seguenti:

V_{Dmax} = 25V a circuito aperto,
C_{Dmax} = non specificata,
R_D = Il suo valore deve essere tale che un eventuale corto circuito tra due conduttori non generi una corrente maggiore di 0,5 A,
dv/dt = compreso tra 6V/ms e 30V/μs (per il *driver),*
E_{Rmax} = 2V.

TAVOLA 1.

Connessioni e segnali assegnati alle interfacce EIA RS-232-C e CCITT V. 24.

PIN del connettore	*Circuito EIA*	*Circuito CCITT*	*Descrizione*	*Terra*	*Dati*		*Controlli*		*Clock*	
					dal DCE	*al DCE*	*dal DCE*	*al DCE*	*dal DCE*	*al DCE*
1	*A A*	1 0 1	Terra protettiva	x						
2	*B A*	1 0 3	Dati da trasmettere			x				
3	*B B*	1 0 4	Dati ricevuti		x					
4	*C A*	1 0 5	Richiesta di trasmissione					x		
5	*C B*	1 0 6	Pronto a trasmettere				x			
6	*C C*	1 0 7	*DCE* pronto				x			
7	*A B*	1 0 2	Terra del segnale (ritorno comune)	x						
8	*C F*	1 0 9	Rivelazione del segnale di linea				x			
9	—		Riservato per controllo							
10	—		Riservato per controllo							
11	—		Non assegnato							
12	*S C F*	1 2 2	Rivelazione segnale secondario							
13	*S C B*	1 2 1	Pronto a trasmettere secondario				x			
14	*S B A*	1 1 8	Trasmissione dati secondari			x				
15	*D B*	1 1 4	Clock del trasmettitore (*DCE*)						x	
16	*S B B*	1 1 9	Ricezione dati secondari		x					
17	*D D*	1 1 5	Clock del ricevitore (*DCE*)						x	
18	—		Non assegnato							
19	*S C A*	1 2 0	Richiesta di trasmissione secondaria					x		
20	*C D*	1 0 8.2	Termine lista dati					x		
21	*C G*	1 1 0	Rivelazione di qualità del segnale				x			
22	*C E*	1 2 5	Indicatore di chiamata				x			
23	*C H / C I*	1 1 1 / 1 1 2	Selettore velocità di trasmissione (*DTE/DCE*)				x	x		
24	*D A*	1 1 3	Clock del trasmettitore (*DTE*)							x
25	—		Non assegnato							

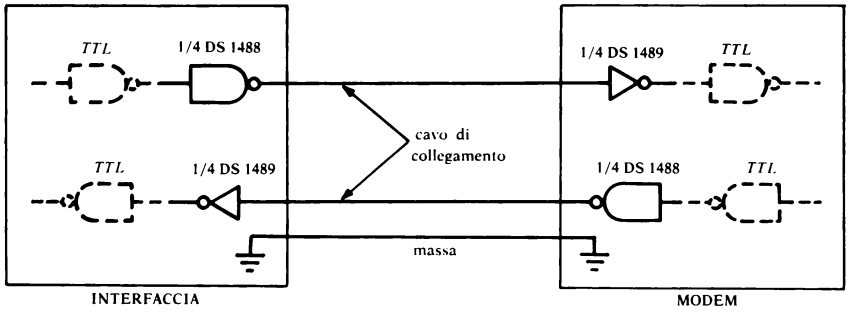
R_R = da tre a sette KΩ,
C_{Rmax} = 2500pF.

Il limite posto alla capacità del carico impone delle ovvie limitazioni alla lunghezza del cavo di collegamento se non si fa uso di *modem.*

Se la linea viene pilotata alla massima velocità (circa 20Kbaud), la lunghezza massima ammissibile si aggira intorno a poche decine di metri che però possono diventare molti di più scendendo fino a velocità dell'ordine dei 1200 baud.

Anche per queste interfacce esistono degli integrati che hanno la funzione di convertire i livelli logici *TTL* o *CMOS* in quelli appropriati per lo standard *EIA* e *CCITT* e viceversa. Come esempio si può citare il DS 1488 della National che è un pilota di linea quadruplo, ed il DS 1489 che è un ricevitore di linea quadruplo, adatti per livelli *TTL*.

Nella fig. 14 è mostrata una tipica applicazione nella trasmissione dati con alcune delle principali caratteristiche.



	DS 1488	DS 1489
tensione d'alimentazione	$V^+ = +15V$ $V^- = -15V$	10 V
tensione d'ingresso (V_{in})	$-15V \leqq V_{in} \leqq 7V$	$-30V \leqq V_{in} \leqq +30V$
tensione d'uscita	15V	typ. 3,8V
corrente d'uscita	12mA	20mA

FIG. 14. - Esempio di conversione da TTL allo standard EIA

Loop di corrente

Questo tipo di interfaccia, inizialmente adottato nei sistemi telegrafici, viene utilizzato per la trasmissione di dati seriali asincroni *half-duplex* e *full-duplex* ed ha il grande vantaggio di una notevole semplicità e versatilità. Per contro, i suoi parametri caratteristici non sono perfettamente definiti e standardizzati per cui possono sorgere delle difficoltà per il progettista.

Nella fig. 15 è mostrato lo schema di principio di un collegamento in *loop* di corrente a 20mA.

In questo caso i livelli di tensione, ai quali erano abbinate le informazioni binarie 0 ed 1, vengono sostituiti dalla presenza o assenza della corrente. Ad esempio, il livello logico 1 coincide con la presenza della corrente, mentre l'assenza della corrente individua uno zero logico.

Il valore di 20mA è stato inizialmente scelto poiché era il minimo che garantiva un contatto elettrico sicuro nei dispositivi elettromeccanici, i quali erano costituiti da veri e propri interruttori nella parte trasmittente e da relé in quella ricevente. Attualmente questi dispositivi sono stati sostituiti da dispositivi a stato solido quali i disaccoppiatori optoelettronici.

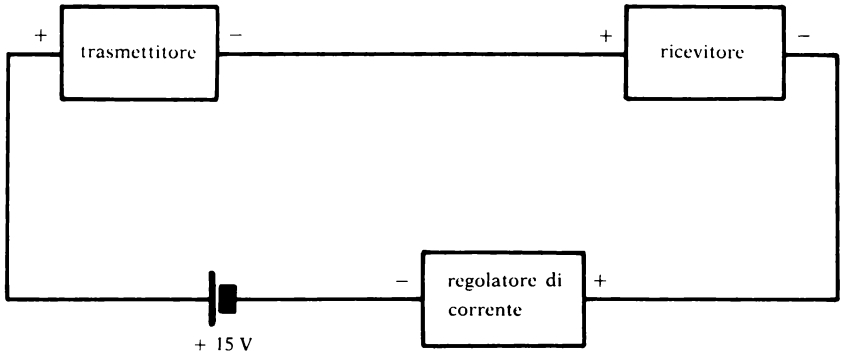


FIG. 15. - Principio di funzionamento del loop di corrente.

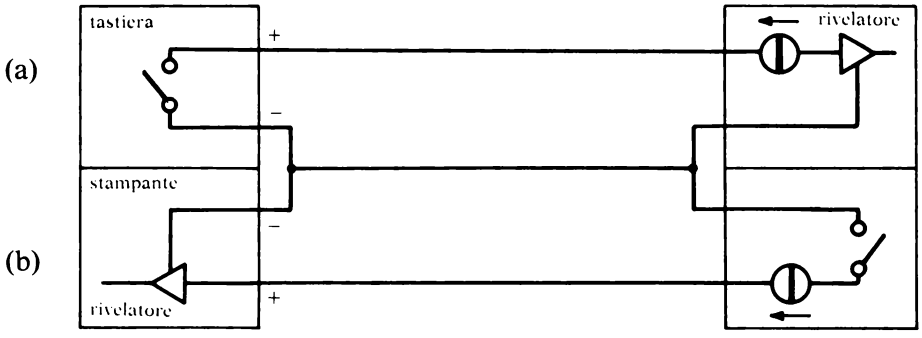


FIG. 16. - (a) loop di tastiera; (b) loop di stampante.

Nei sistemi di tipo *full-duplex* sono necessari due *loop* di corrente per effettuare la comunicazione simultaneamente in entrambi i sensi, che vengono chiamati *loop* di tastiera e *loop* di stampante (fig. 16).

Naturalmente, se la tensione di alimentazione è di 15V, la resistenza totale del circuito per ottenere 20mA deve essere di 750 Ω.

Supponendo che le resistenze interne dei dispositivi ammontino totalmente a 600 Ω, i restanti 150 Ω sono da sfruttarsi per la linea di trasmissione che, ad esempio, per un cavo di rame del diametro di 0,5 mm presenta una resistenza di 88 ohm/Km, equivalente a 1700 metri.

Poiché si lavora in *loop* il ricevitore può essere posto ad una distanza di 850 metri dal trasmettitore (sempreché non sopravvengano altri tipi di considerazioni e di disturbi).

Nei circuiti telegrafici, dove le distanze sono notevolmente superiori, si utilizzano tensioni più elevate (120 o 260 V) e dispositivi con resistenze interne più piccole, per permettere lunghezze superiori dei cavi di collegamento.

Il generatore di corrente può essere localizzato o nel trasmettitore o nel ricevitore e viene chiamato attivo il dispositivo che lo contiene, mentre viene chiamato passivo l'altro. Risulta chiaro che il collegamento diretto può realizzarsi solamente tra un terminale attivo ed uno passivo e non fra due entrambi attivi oppure entrambi passivi.

11. Generatori di Baude-Rate

Nei sistemi a microprocessore accade spesso di aver bisogno di un clock a frequenze standard per il funzionamento di componenti specifici di supporto quali per esempio il già visto USART.

Le frequenze richieste di solito si ricavano dividendo opportunamente la frequenza di clock del sistema mediante normali integrati TTL oppure per mezzo di componenti particolari dedicati conosciuti con il nome di generatori di baude-rate.

Uno dei più economici e diffusi generatori di baude-rate è il 4702 della Fairchild. Il 4702 è capace di generare uno qualsiasi dei 14 bit rates comunemente usati mediante un oscillatore a quarzo esterno collegato all'ingresso di clock (fig. 17).

Nelle fig. 18 e 19 sono riportate due tipiche applicazioni di questo integrato.

Il circuito di fig. 18 fornisce una tra le cinque possibili bit rates a seconda della posizione dello switch. L'uscita di bit rate (z) è in grado di guidare un carico normale TTL o quattro carichi Schottky. Le frequenze di uscita

4702/4702B

PROGRAMMABLE BIT-RATE GENERATOR

FAIRCHILD CMOS MACROLOGIC

DESCRIPTION - The 4702 Bit-Rate Generator provides the necessary clock signals for digital data transmission systems, such as Universal Asynchronous Receiver and Transmitter circuits (UARTs). It generates any of the 14 commonly used bit rates using an on-chip crystal oscillator, but its design also provides for easy and economical multi-channel operation, where any of the possible frequencies must be made available on any output channel.

One 4702 can control up to eight output channels. When more than one bit-rate generator is required, they can still be operated from one crystal.

- **PROVIDES 14 COMMONLY USED BIT-RATES**
- **ONE 4702 CONTROLS UP TO EIGHT TRANSMISSION CHANNELS**
- **USES 2.4576 MHz INPUT FOR STANDARD FREQUENCY OUTPUTS (16 TIMES BIT RATE)**
- **CONFORMS TO EIA RS-404**
- **ON-CHIP INPUT PULL UP CIRCUITS**
- **TTL COMPATIBLE-OUTPUTS WILL SINK 1.6 mA**
- **INITIALIZATION CIRCUIT FACILITIES DIAGNOSTIC FAULT ISOLATION**
- **LOW POWER DISSIPATION - 1.35 mA POWER DISSIPATION AT 5 V AND 2.4576 MHz**
- **16-PIN DUAL IN-LINE PACKAGE**

TABLE 1

CLOCK MODES AND INITIALIZATION

I_X	$\bar{E}_{CP}$	CP	OPERATION
⎍⎍⎍	H	L	Clocked from I_X
X	L	⎍⎍⎍	Clocked from CP
X	H	H	Continuous Reset
X	L	⎍⎍	Reset During First CP = HIGH Time

H = HIGH Level
L = LOW Level
X = Don't Care
⎍⎍ = 1st HIGH Level Clock Pulse After $\bar{E}_{CP}$ Goes LOW
⎍⎍⎍ Clock Pulses

Note 1: Actual output frequency is 16 times the indicated output rate, assuming a clock frequency of 2.4576 MHz.

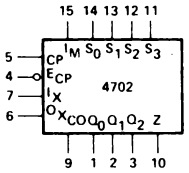
TABLE 2

TRUTH TABLE FOR RATE SELECT INPUTS

S_3	S_2	S_1	S_0	Output Rate (Z) Note 1
L	L	L	L	Multiplexed Input (I_M)
L	L	L	H	Multiplexed Input (I_M)
L	L	H	L	50 Baud
L	L	H	H	75 Baud
L	H	L	L	134.5 Baud
L	H	L	H	200 Baud
L	H	H	L	600 Baud
L	H	H	H	2400 Baud
H	L	L	L	9600 Baud
H	L	L	H	4800 Baud
H	L	H	L	1800 Baud
H	L	H	H	1200 Baud
H	H	L	L	2400 Baud
H	H	L	H	300 Baud
H	H	H	L	150 Baud
H	H	H	H	110 Baud

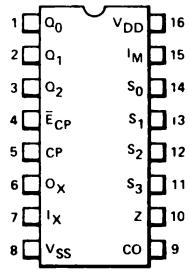
FIG. 17. - Tabella e piedinatura del 4702 (dal manuale tecnico Fairchild)

LOGIC SYMBOL



V_{DD} = Pin 16
V_{SS} = Pin 8

CONNECTION DIAGRAM
DIP (TOP VIEW)



NOTE:
The Flatpak version has the same pin-outs (Connection Diagram) as the Dual In-line Package.

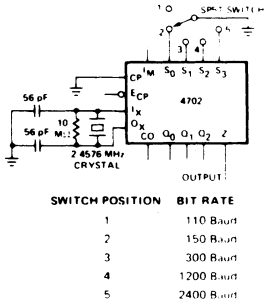
PIN NAMES

CP	External Clock Input
$\bar{E}_{CP}$	External Clock Enable Input (Active LOW)
I_X	Crystal Input
I_M	Multiplexed Input
S_0-S_3	Rate Select Inputs
CO	Clock Output
O_X	Crystal Drive Output
Q_0-Q_2	Scan Counter Outputs
Z	Bit Rate Output

ottenibili corrispondono a 110, 150, 300, 1200 e 2400 Baud, più che sufficienti per terminali a basso costo.

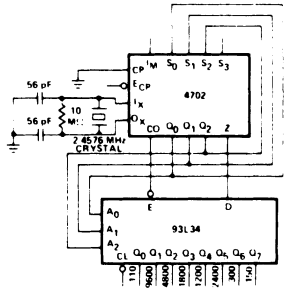
Il circuito di fig. 19 è invece capace di generare otto bit rates su otto linee di uscita utilizzando un 4702 e un 93L34 (latch ad otto bit indirizzabile).

Se la programmazione della frequenza desiderata si volesse effettuare via software anziché via hardware, occorre servirsi di altri componenti LSI più sofisticati quali ad esempio l'8253 (Programmable Interval Timer) della INTEL (simile al CTC Z 80) che tra le varie funzioni svolte ha pure la caratteristica di poter essere usato come generatore di baude-rate programmabile.



SWITCH SELECTABLE BIT RATE GENERATOR CONFIGURATION PROVIDING FIVE BIT RATES

FIG. 18.



BIT RATE GENERATOR CONFIGURATION WITH EIGHT SIMULTANEOUS FREQUENCIES

FIG. 19.

12. Rumore

In tutta la precedente trattazione si è sempre tacitamente supposto che le informazioni, sotto forma di 1 e di 0, rispettassero la configurazione ideale. Nella pratica invece, le correnti o le tensioni che costituiscono il bit, non sono sempre così esattamente corrispondenti alla loro espressione matematica. Una delle principali cause è il rumore (*noise).*

Il rumore può essere esterno al sistema digitale o interno.

La causa del primo può essere individuata negli apparati che danno origine a perturbazioni elettromagnetiche quali: motori elettrici, relays, telescriventi, saldatrici ad arco, ecc. ed arrivano al sistema attraverso le linee di alimentazione o dei dati.

Sulla linea di rete le interferenze che possono causare malfunzionamenti possono raggrupparsi in tre tipi:

a) sovratensioni di ampiezza fino a 2 KV con tempi di salita dell'ordine di 1 μs e durata di circa 50 μs.

b) impulsi ripetitivi con tempi di salita di pochi ns e durata fino ad 1 μs.
c) mancanza della tensione di linea per un massimo di 60 μs.

L'esperienza ha dimostrato che poiché le cause del rumore esterno sono raramente connesse in modo diretto alla logica, la loro influenza è generalmente trascurabile con un buon progettato alimentatore. In alcuni casi particolari, che vengono indicati direttamente dal costruttore, come mezzi di protezione si può ricorrere ai seguenti mezzi:
1) schermatura;
2) filtraggio nel punto dove il cavo entra nell'apparecchiatura;
3) corretta disposizione delle masse;
4) accurata disposizione dei circuiti logici.

Il rumore interno è invece costituito essenzialmente dai rimbalzi di linea e dall'accoppiamento elettromagnetico tra conduttori che prende il nome di *crosstalk*.

Nei successivi paragrafi si studierà il solo rumore interno prendendo in considerazione, per semplicità, solo le linee con carichi concentrati all'inizio ed alla fine. Per lo studio di situazioni più complesse diventa indispensabile ricorrere alla simulazione per mezzo di un elaboratore se si vuole raggiungere una analisi completa e dettagliata.

Rimbalzi

La durata massima di un rumore che può essere prodotto su di un filo è di 1 ÷ 1,3 ns ogni 10 cm, per cui si può dedurre per ogni famiglia logica la massima lunghezza dei collegamenti per non incorrere in inconvenienti dovuti al rumore stesso.

Il problema non è di così facile soluzione poiché mentre, ad esempio, per un flip-flop di tipo TTL che abbia un tempo minimo di setup di 10 ns si può arrivare a lunghezze dell'ordine di 10·10 cm/1,3 ns = 80 cm; con famiglie veloci come la TTL Schottky o la ECL che hanno tempi di assestamento di pochi ns, occorrerebbero dei collegamenti al di sotto di 1 ÷ 2 cm.

Nessuna preoccupazione sorge invece, usando famiglie lente come la C-MOS, tranne i casi in cui sia necessaria una vera e propria linea di trasmissione per comunicare dati ad elevate distanze e per le quali sono impiegate particolari apparecchiature.

Va tenuto in ogni caso presente che gli impulsi di rumore, se hanno una durata inferiore al tempo di assestamento (set up) della porta logica che li riceve, non riescono a modificarne lo stato di uscita e sono perciò trascurabili.

Tutti i fili di connessione fra elementi circuitali possono considerarsi come linee di trasmissione e come tali caratterizzati da una velocità di propagazione e da una impedenza caratteristica. I parametri principali che ne influenzano il valore sono: le terminazioni, i carichi capacitivi, la lunghezza delle linee che si dipartono da quella principale (stub), gli accoppiamenti con i connettori vicini.

Il tempo di propagazione lungo le piste di un circuito stampato, è di circa 6 ns/m ma viene aumentato dalle capacità parassite degli ingressi delle porte logiche ad esse connesse e può essere calcolato mediante l'equazione:

$$t'_{pd} = t_{pd} \cdot \sqrt{1 + \frac{C_D}{C_0 \cdot l}}$$

dove t_{pd} è il tempo di propagazione della linea non caricata, C_D è la capacità totale distribuita dovuta alle porte logiche, C_0 è la capacità intrinseca della linea per unità di lunghezza, ed l è la lunghezza della linea.

L'impedenza caratteristca Z_0 (vedi «Elettronica Digitale»), normalmente rientra tra i 30 Ω ed i 600 Ω e può considerarsi di circa 100 Ω per una connessione di tipo *wire-wrap*. Se la linea è caricata, l'impedenza caratteristica si abbassa notevolmente secondo l'equazione:

$$Z'_0 = \frac{Z_0}{\sqrt{1 + \frac{C_D}{C_0 \cdot l}}}$$

dove Z_0 è l'impedenza della linea non caricata, C_D è la capacità totale distribuita comprensiva dei carichi, C_0 è la capacità intrinseca della linea per unità di lunghezza ed infine l è la lunghezza della linea.

ESEMPIO

Una pista di circuito stampato ha una lunghezza l = 50 cm, una impedenza caratteristica intrinseca Z_0 = 100 Ω, un tempo di propagazione t_{pd} = 6 ns/m ed una capacità intrinseca di 30 pF/m.

Se la pista viene caricata con 15 *line Driver-Receivers* DS 7838, come mostrato nella figura 20, i quali hanno verso la linea una capacità di 15 pF

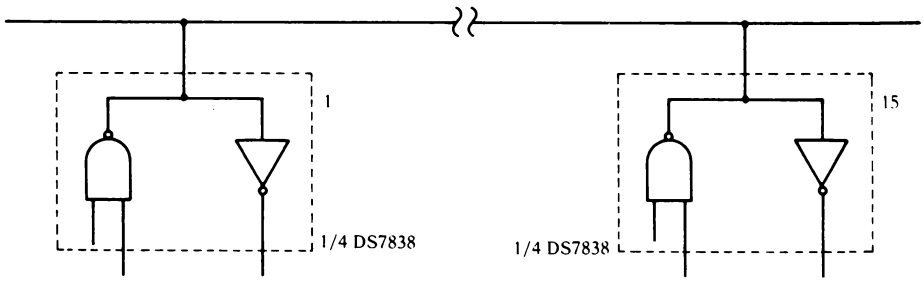


FIG. 20. - Linea caricata con 15 piloti/ricevitori di linea.

(la capacità di ingresso del *receiver* più quella di uscita del driver), il nuovo valore dell'impedenza caratteristica risulta:

$$Z'_0 = \frac{100}{\sqrt{1 + \frac{225\ pF}{15\ pF}}} = 25\ \Omega$$

mentre il tempo di propagazione vale:

$$t'_{pd} = 6\ 10^{-9} \cdot 0{,}5 \sqrt{1 + \frac{225\ \ 10^{-9}}{15\ \ 10^{-9}}} = 12\ ns/m$$

Se la linea non è chiusa su un'impedenza uguale a quella caratteristica, si otterranno delle riflessioni la cui entità è data dal prodotto del segnale in arrivo per il coefficiente di riflessione che è espresso dalla relazione:

$$K = \frac{Z_L - Z_0}{Z_L + Z_0}$$

dove con Z_0 è indicata l'impedenza caratteristica e con Z_l quella sul carico.

La conoscenza di questo valore (o del suo equivalente ρ), permette di tracciare in prima approssimazione l'andamento della tensione o della corrente alle due estremità (o lungo il percorso) di una linea di trasmissione.

Si cercherà di chiarirne il significato mediante un esempio.

Si supponga di collegare un generatore di f.e.m continua di valore E_s e

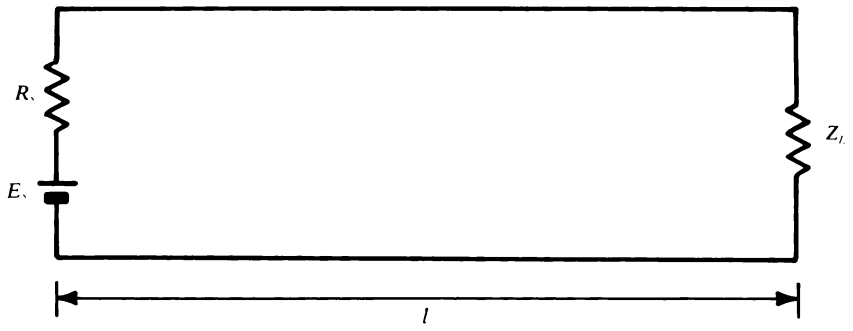


FIG. 21. - Linea di trasmissione con ingresso a gradino.

resistenza interna R_s ad un estremo di una linea di lunghezza *l*, , chiusa all'altro estremo su una impedenza Z_L diversa da quella caratteristica (fig. 21).

Se all'istante t = 0 viene connesso il generatore alla linea, fino a quando il gradino di tensione non raggiunge il carico, l'impedenza mostrata dalla linea verso il generatore è Z_0 (impedenza di una linea di lunghezza infinita), per cui l'ampiezza della tensione che viaggia lungo di essa, a velocità paragonabile a quella della luce, sarà:

$$E_1 = \frac{E_s \cdot Z_0}{R_s + Z_0}$$

Dopo un tempo pari a *l/u* (*u* = velocità della luce), questo gradino raggiunge il carico dove subisce una riflessione parziale pari al valore del coefficiente K_L sul carico, ottenendo in tal modo un'onda che viaggia dal carico verso il generatore, con ampiezza:

$$E_2 = K_L E_1$$

Dopo un tempo pari a 2l/u questo segnale giungerà al generatore e se la sua impedenza (R_s) è diversa da quella caratteristica si otterrà un'ennesima riflessione con coefficiente K_S, e via di seguito.

Tale comportamento è mostrato dal diagramma delle riflessioni di fig. 22 mentre nella fig. 23 sono mostrate le forme d'onda che ne derivano.

Questi andamenti sono stati ottenuti sommando algebricamente le tensioni alle estremità trasmittente $(d = 0)$ e ricevente $(d = l)$.

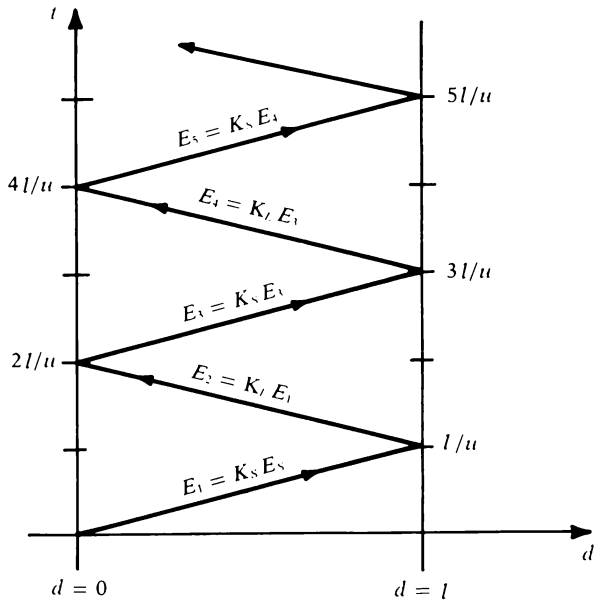


FIG. 22. - Diagramma delle riflessioni in una linea non adattata.

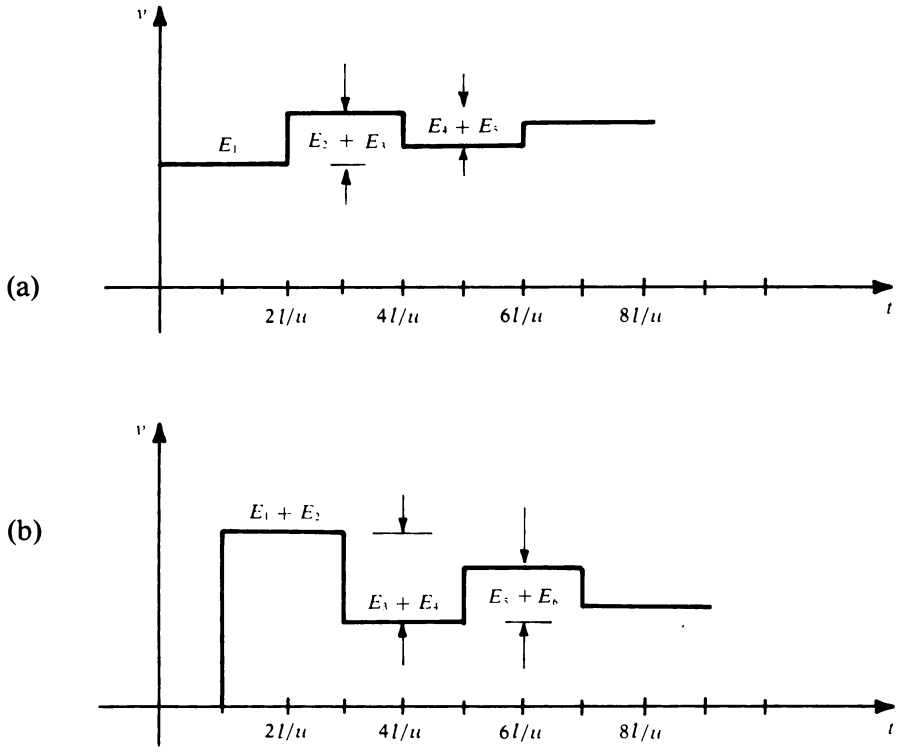


FIG. 23. - Tensioni all'estremità trasmittente (a) e ricevente (b) con ingresso a gradino.

Purtroppo però, molti dei circuiti di interesse in elettronica digitale hanno delle caratteristiche tensione/corrente di tipo non lineare per cui viene introdotta una nuova tecnica che permetterà di leggere con facilità direttamente le tensioni a ciascuna estremità della linea.

Essa consiste nella soluzione, per via grafica, del sistema formato dalle equazioni relative alle terminazioni ed alla linea di trasmissione. Nella figura 24-b è mostrato il metodo di soluzione relativo al circuito di figura 24-a.

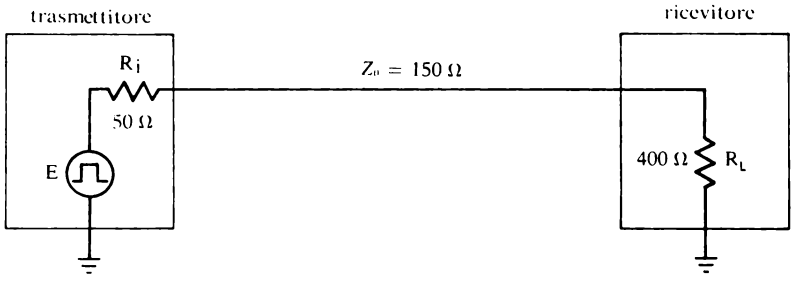


FIG. 24 a - Linea di trasmissione disadattata.

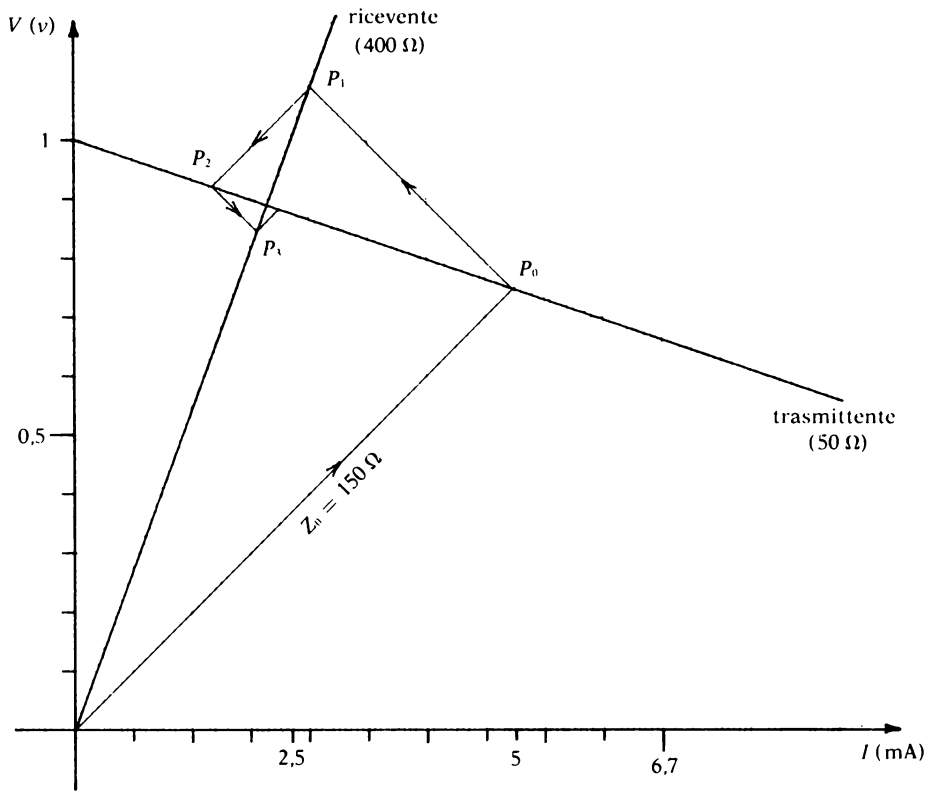


FIG. 24 b - Soluzione grafica per il calcolo dei rimbalzi.

Su un sistema di assi cartesiani V - I, la scala viene scelta (per comodità, ma non è necessario) in modo che ogni linea inclinata di 45° rappresenti un valore di corrente e di tensione il cui rapporto sia pari all'impedenza caratteristica della linea (nel caso precedente $V/I = Z_0 = 150\ \Omega$, per cui ad 1 V corrisponde un valore di corrente pari ad $1/150 = 6{,}6$ mA).

Scelta la scala, vanno disegnate le due rette di carico relative all'estremità ricevente ed a quella trasmittente, che rappresentano tutte le possibili combinazioni di valori V-I, che vi possono formarsi. Poiché sono sufficienti due punti per tracciare una retta, per l'estremità ricevente si ha:

per $V = 0$, si ha $I = 0$

per $V = 1$, si ha $I = \dfrac{1}{400} = 2{,}5$ mA

mentre per quella trasmittente:

per V = 1, si ha I = 0

per V = 0, si ha I = $\frac{1}{50}$ = 20 mA

I reali valori alle due estremità della linea, costituiti dalla soluzione del sistema, si ottengono alle intersezioni delle rette di carico con quella rappresentante l'impedenza caratteristica.

Partendo dall'origine, con una inclinazione di 45° (pendenza pari al valore di Z_0), si raggiunge la retta di carico nel punto P_0 cui corrispondono i valori iniziali: I = 5 mA, e V = 0,75 V per l'estremità trasmittente.

Dal punto P_0, con una retta anch'essa inclinata di 45°, si raggiunge il punto P_1 che dà come valori: I = 2,7 mA e V = 1,09 V per l'estremità ricevente. Continuando allo stesso modo, si ottengono i valori relativi ai punti P_2, P_3, ecc. fino ad una sufficiente approssimazione al valore finale cui tendono (incrocio delle due rette di carico).

Trasferendo i valori di tensione su un diagramma in funzione del tempo, si ottengono le forme d'onda di figura 24, che mostrano come viene deformato un gradino inviato su di una linea non adattata.

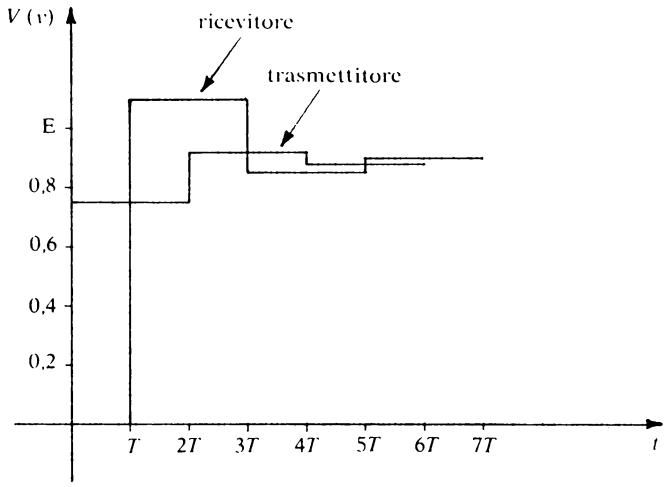


FIG. 25. - Deformazione del gradino alle estremità dela linea di figura 23a.

13. Effetto delle riflessioni su un sistema digitale

La grande potenza del precedente metodo grafico di soluzione, viene sfruttata quando la linea in esame è il collegamento tra due circuiti integrati digitali le cui rette di carico sono non lineari. Nella figura 26 sono mostrate le caratteristiche di ingresso ed uscita per i livelli alti e bassi di

TYPICAL CHARACTERISTICS†§

OUTPUT VOLTAGE
vs
INPUT VOLTAGE

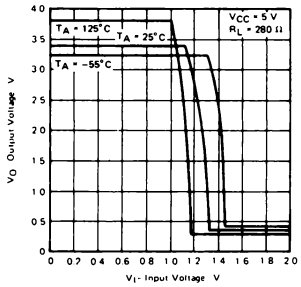


FIGURE E1

INPUT-CLAMPING-DIODE
FORWARD VOLTAGE
vs
FREE-AIR TEMPERATURE

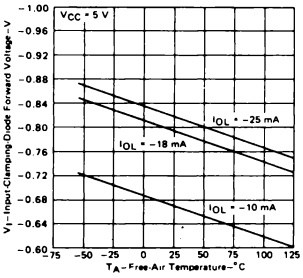


FIGURE E2

HIGH-LEVEL OUTPUT VOLTAGE
vs
HIGH-LEVEL OUTPUT CURRENT

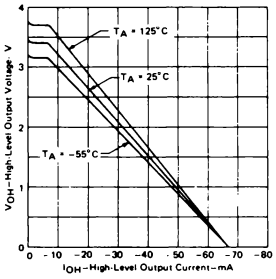


FIGURE E3

LOW-LEVEL OUTPUT VOLTAGE
vs
LOW-LEVEL OUTPUT CURRENT

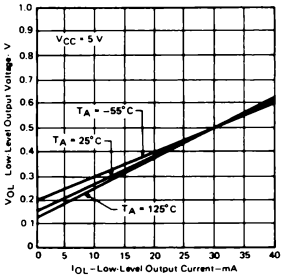


FIGURE E4

INPUT CURRENT
vs
INPUT VOLTAGE

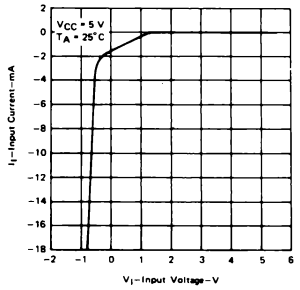


FIGURE E5

HIGH-LEVEL INPUT CURRENT
vs
FREE-AIR TEMPERATURE

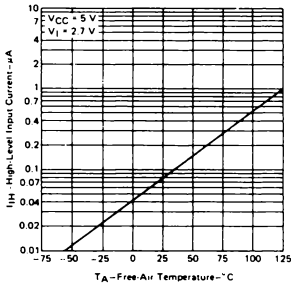


FIGURE E6

† Data for temperatures below 0°C and above 70°C are applicable for Series 54S circuits only.
§ Data as shown are applicable specifically for the NAND gates with totem-pole outputs.

FIG. 26. - Caratteristiche tipiche della famiglia TTL Schottky (Texas Instruments).

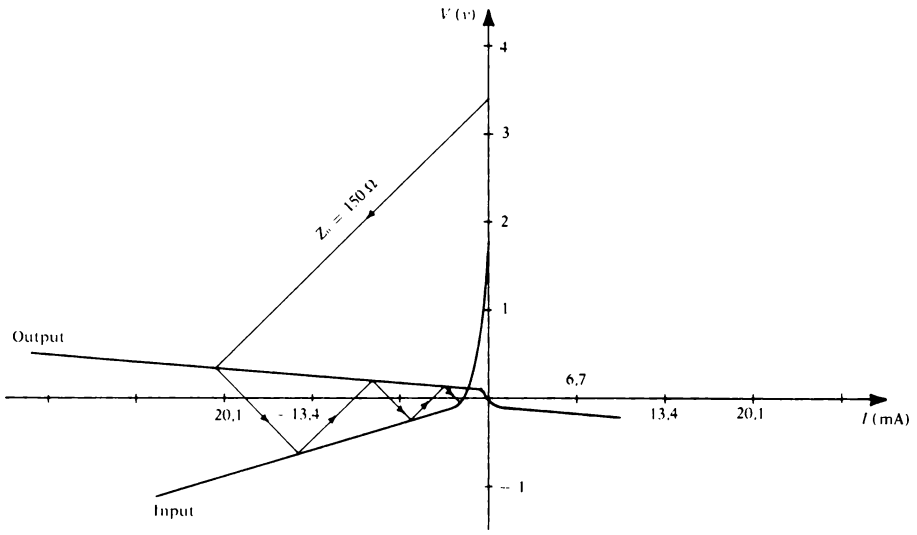


FIG. 27. - Rimbalzi TTL nella transizione da alto a basso.

una porta TTL. (Da notare che è necessario riportare anche i quadranti con tensioni negative poiché i rimbalzi possono far diventare negativo un segnale in ingresso).

Volendo effettuare una verifica sull'entità dei rimbalzi nella transizione da alto a basso in un collegamento tramite una linea di impedenza $Z_0 = 150\ \Omega$, occorre riportare in un unico grafico le caratteristiche E4 ed E5 di figura 26. Con analogo procedimento a quanto visto per carichi resistivi, si ottiene il grafico di figura 27.

Da esso è possibile osservare come nel caso attuale le riflessioni siano di modesta entità e soprattutto non si originano picchi positivi successivi

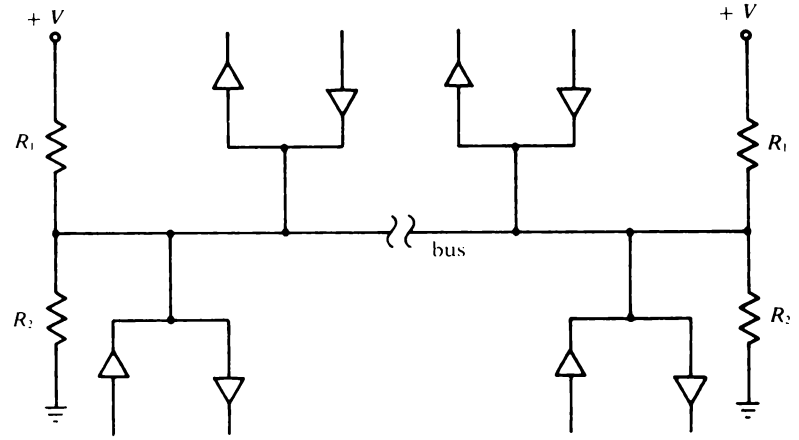


FIG. 28. - Tipica configurazione di un bus terminato.

che, nel caso di segnali di clock, possono far scambiare un singolo impulso come due o più impulsi.

Nelle linee organizzate a bus, lo scambio dei dati avviene tramite *driver* e *receiver* connessi in vari punti casuali del bus (figura 28).

Se la lunghezza dei collegamenti è limitata all'interno di una cartolina standard, non è necessario terminare la linea, mentre negli altri casi la terminazione va effettuata mediante l'inserimento dei resistori R_1 ed R_2 calcolati in modo tale che il loro parallelo sia uguale all'impedenza caratteristica Z_0' della linea e siano rispettate le esigenze di fan-out in d.c. Per fare ciò si può anche disadattare parzialmente la linea. Ad esempio per una impedenza caratteristica pari a 50 Ω i valori potrebbero essere: R_1 = 180 Ω ed R_2 = 300 Ω, pari al doppio di Z_0 per esigenze di fan-out.

14. Rumore sulle linee di alimentazione

Nelle figure 29a e 29b, sono mostrati i risultati delle prove sperimentali sull'immunità al rumore dinamico relative alla linea di alimentazione e di massa, per diverse famiglie logiche.

Esse riportano l'ampiezza minima del rumore (in volt) capace di influenzare il corretto funzionamento di una porta, relativamente alla durata del rumore stesso.

Una delle cause del suo insorgere sulla linea di massa (che anch'essa deve essere considerata una linea di tramissione), sono le rapide variazioni delle correnti che deve sopportare in conseguenza delle variazioni degli stati delle porte.

Si prenda ad esempio il f-f di tipo D di figura 30. L'impulso di clock causa una variazione dell'uscita che inizialmente è solo la metà dell'intera

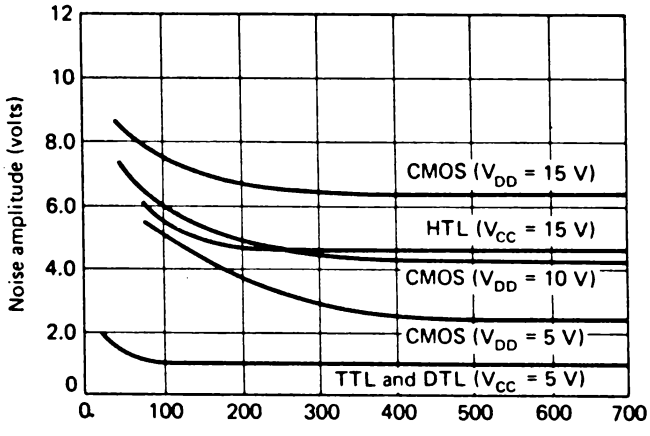


FIG. 29.-

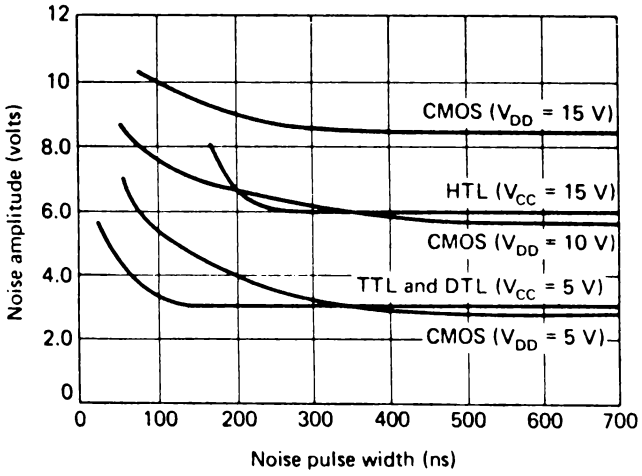


FIG. 29. - Immunità al rumore sulla linea di alimentazione (a) e sulla linea di massa (b). (Motorola).

ampiezza poiché essa si divide tra le impedenze della linea del segnale e della linea di massa. Questo fronte in salita, viaggiando lungo la linea, raggiunge il punto di connessione fisica alla massa e riceve una riflessione con coefficiente pari a -1 (R = 0) che riporta a zero la tensione sul piedino GND del f-f dopo un tempo pari a 2T (2T è il tempo di andata e ritorno dal piedino GND alla massa effettiva).

Per 10 cm di pista di circuito stampato, si ottiene una durata dell'impulso pari a: 20 cm · 0,13 ns/cm = 2,6 ns.

Ne risulta che l'effettiva forma d'onda della tensione applicata tra l'ingresso di clock e massa è come quella a tratto intero in figura e che può essere erroneamente considerato come due impulsi di clock successivi nelle logiche più veloci.

Per eliminare questo inconveniente si possono congiungere insieme le masse del Ck e delle porte logiche come mostrato nella figura 31. Però in tal modo, se è vero che il f-f A è immune al rumore che compare contemporaneamente sul clock e sulla massa dato che la d. di p. non cambia, sarà il f-f B ad essere affetto dal rumore sull'ingresso CK ma non sulla massa con conseguente possibilità di malfunzionamento.

Un altro metodo generalmente utilizzato è quello di ridurre l'impedenza caratteristica delle linee utilizzando piastre per circuiti stampati con una faccia interamente connessa a massa (mother-board) oppure allargando il più possibile le zone di rame ad essa connesse.

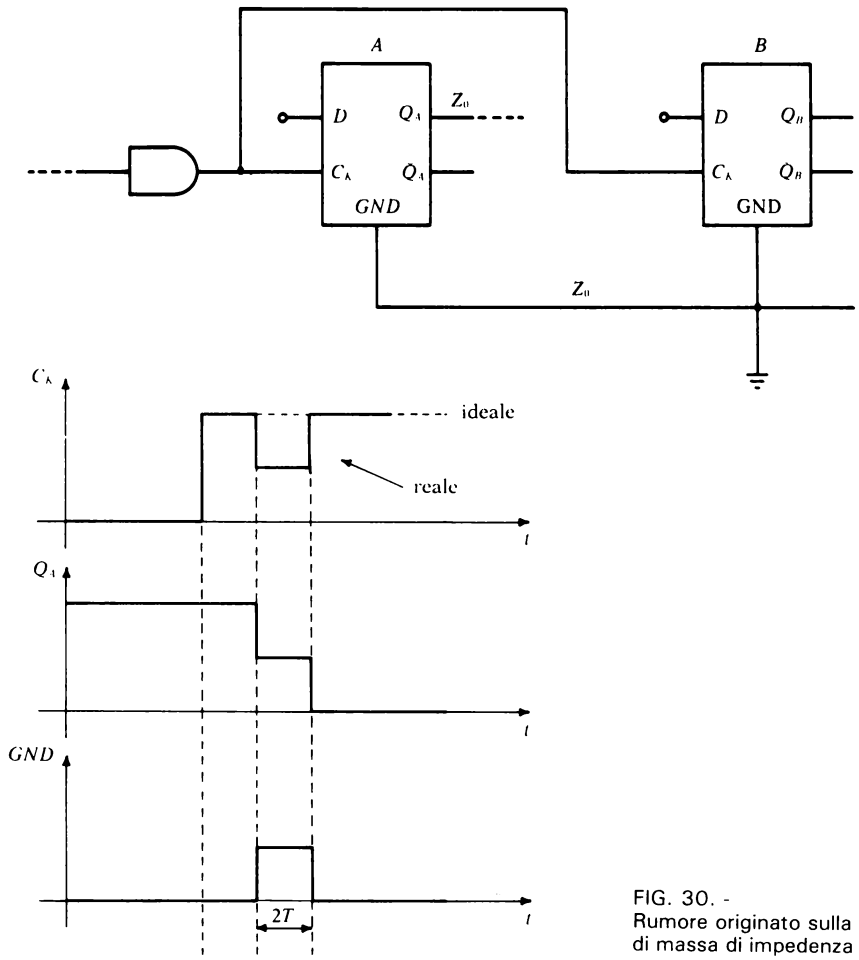


FIG. 30. -
Rumore originato sulla linea di massa di impedenza Z_0.

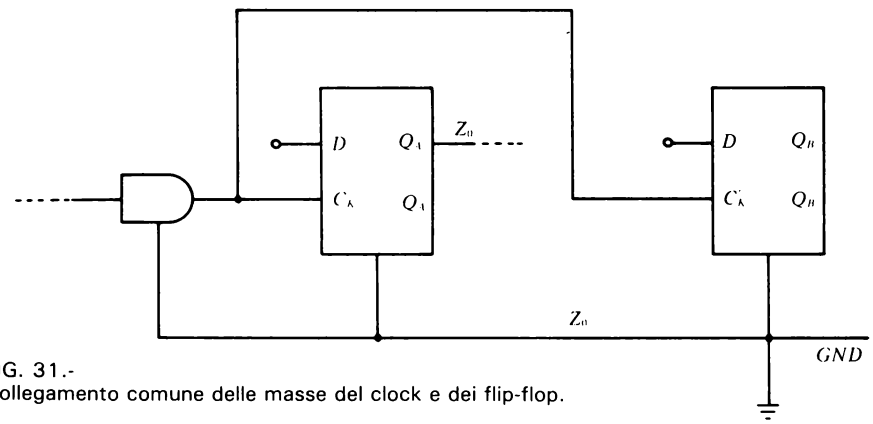


FIG. 31.-
Collegamento comune delle masse del clock e dei flip-flop.

15. Crosstalk

Il problema dell'accoppiamento induttivo e capacitivo tra le linee (*crosstalk)*, sorge solamente quando si utilizzano porte veloci (tipo ECL) funzionanti a velocità prossime a quella massima, oppure quando si debbono trasmettere informazioni a distanza mediante l'uso dei cavi. Usando connettori piatti a doppio dielettrico o cavi coassiali, si può operare su distanze fino a circa 7 ÷ 8 metri usando normali porte logiche. Per distanze maggiori, il metodo generalmente più utilizzato è quello della trasmissione differenziale (figura 32) che riduce notevolmente sia il rumore di massa che il crosstalk.

I valori di resistenza di R_1, R_2, R_3 ed R_4 vengono indicati di volta in volta sui manuali, relativamente ai dispositivi adoperati.

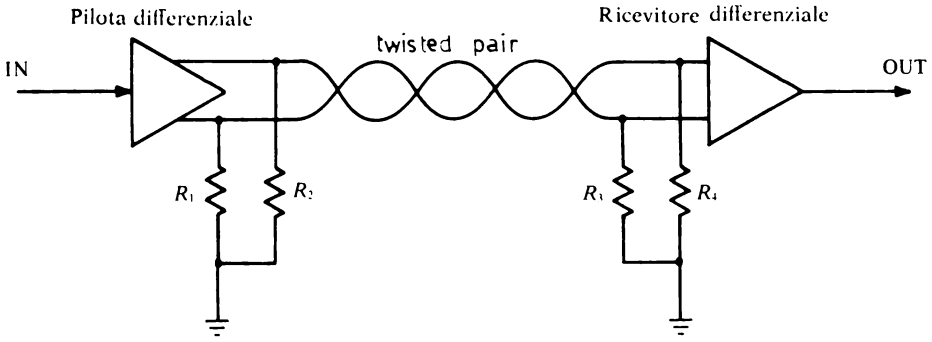


FIG. 32. - Sistema di trasmissione differenziale.

La massima lunghezza possibile dei collegamenti, realizzati tramite coppie di fili intrecciati, può essere calcolata facendo il rapporto fra la massima attenuazione ammissibile del cavo (A_M) e quella per unità di lunghezza (A_u):

$$l_{max} = \frac{A_M}{A_u}$$

ESEMPIO

Una linea è realizzata tramite una coppia di fili intrecciati di impedenza caratteristica $Z_0 = 75\ \Omega$, che presenta alla frequenza di lavoro di 10 MHz, una attenuazione di 7,5 dB per 100 metri di lunghezza.

Come pilota e ricevitore di linea sono stati utilizzati un SN 75110 ed un SN 75107 rispettivamente. Nella figura 33 è mostrato lo schema delle connessioni.

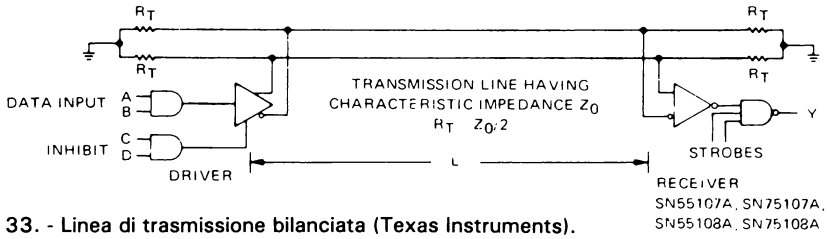


FIG. 33. - Linea di trasmissione bilanciata (Texas Instruments).

La massima lunghezza raggiungibile dai collegamenti può essere calcolata tramite la espressione:

$$l_{max} = \frac{A_M}{A_u}$$

dove A_u vale 7,5 dB/100m, mentre A_M viene calcolata facendo il rapporto tra la tensione differenziale di ingresso alla linea e quella minima necessaria per superare la soglia del ricevitore.

Se le resistenze di terminazione valgono $Z_0/2$, dalle caratteristiche del driver e del receiver riportate parzialmente nelle figure 34 e 35, si ottiene che l'ampiezza del segnale all'ingresso della linea vale:

$$V_{DIFF} = \frac{1}{2} \cdot I_{0\,(on)} \cdot R_T$$

Poichè per la connessione di figura 33, R_T vale $Z_0/2$ si ottiene:

$$V_{DIFF} = \frac{12 \cdot 10^{-3} \cdot 75}{4} = 225 \text{ mV}$$

La minima tensione differenziale rilevabile in ingresso dal ricevitore vale 25 mV, ma per ragioni di sicurezza conviene prendere un valore pari al doppio, per cui la massima attenuazione permessa vale:

$$A_M = \frac{225 \cdot 10^{-3}}{50 \cdot 10^{-3}} = 4,5 = 13 \text{ dB}$$

La massima lunghezza ammessa per la linea vale allora:

$$l_{max} = \frac{A_M}{A_u} = \frac{13 \text{ dB}}{7,5/100 \text{ m}} = 173 \text{ m.}$$

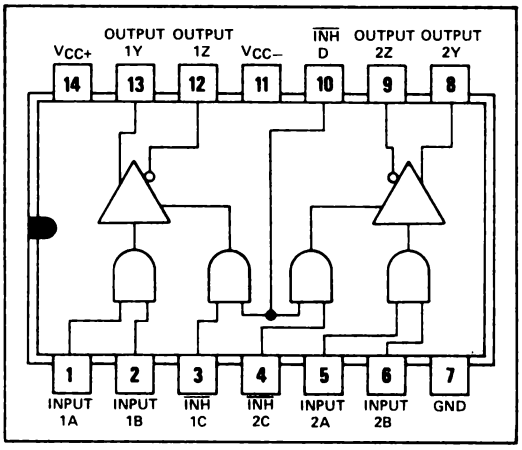
TYPES SN55109A, SN55110A, SN75109A, SN75110A, SN75112 DUAL LINE DRIVERS

BULLETIN NO. DL-S 7712334, DECEMBER 1975—REVISED JANUARY 1977

- Improved Stability over Supply Voltage and Temperature Ranges
- Constant-Current Output
- High Speed
- Standard Supply Voltages
- High Output Impedance
- High Common-Mode Output Voltage Range (−3 V to 10 V)
- TTL Input Compatibility
- Inhibitor Available for Driver Selection

−55°C to 125°C J Package	0°C to 70°C J or N Package	OUTPUT FUNCTION
SN55109A	SN75109A	6-mA Current Switch
SN55110A	SN75110A	12-mA Current Switch
	SN75112	27-mA Current Switch

SN55109A, SN55110A . . . J DUAL-IN-LINE PACKAGE
SN75109A, SN75110A, SN75112 . . . J OR N DUAL-IN-LINE PACKAGE
(TOP VIEW)



description

The SN55109A, SN55110A, SN75109A, SN75110A, and SN75112 have improved output current regulation with supply voltage and temperature variations. In addition the higher current of the SN75112 (27 mA) allows data to be transmitted over longer lines. These drivers offer optimum performance when used with the SN55107A, SN55108A, SN75107A, and SN75108A line receivers.

FUNCTION TABLE

LOGIC INPUTS		INHIBITOR INPUTS		OUTPUTS	
A	B	C	D	Y	Z
X	X	L	X	OFF	OFF
X	X	X	L	OFF	OFF
L	X	H	H	ON	OFF
X	L	H	H	ON	OFF
H	H	H	H	OFF	ON

H = high level, L = low level, X = irrelevant

These drivers feature independent channels with common voltage supply and ground terminals. The significant difference between the three drivers is in the output current specification. The driver circuits feature a constant output current that is switched to either of two output terminals by the appropriate logic levels at the input terminals. The output current can be switched off (inhibited) by low logic levels on the inhibit inputs. The output current is nominally 6 milliamperes for the '109A, 12 milliamperes for the '110A, and 27 milliamperes for the SN75112.

The inhibit feature is provided so the circuits can be used in party-line or data-bus applications. A strobe or inhibitor, common to both drivers, is included for increased driver-logic versatility. The output current in the inhibited mode, $I_{O(off)}$, is specified so that minimum line loading is induced when the driver is used in a party-line system with other drivers. The output impedance of the driver in the inhibited mode is very high—the output impedance of a transistor biased to cutoff.

The driver outputs have a common-mode voltage range of −3 volts to 10 volts, allowing common-mode voltage on the line without affecting driver performance.

All inputs are diode clamped and are designed to satisfy TTL-system requirements. The inputs are tested at 2.0 volts for high-logic-level input conditions and 0.8 volt for low-logic-level input conditions. These tests guarantee 400 millivolts of noise margin when interfaced with Series 54/74 TTL.

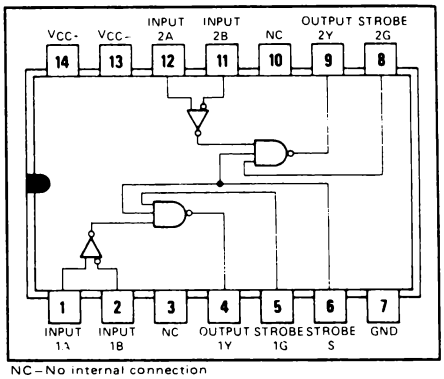
FIG. 34. - Piedinatura e tabella funzionale del doppio pilota di linea SN 75110 (Texas Instruments).

TYPES SN55107A, SN55107B, SN55108A, SN55108B, SN75107A, SN75107B, SN75108A, SN75108B DUAL LINE RECEIVERS

BULLETIN NO. DL-S 7712493, JANUARY 1977

- High Speed
- Standard Supply Voltages
- Dual Channels
- High Common-Mode Rejection Ratio
- High Input Impedance
- High Input Sensitivity
- Differential Input Common-Mode Range of ±3 V
- Differential Input Common-Mode Range of More than ±15 V Using External Attenuator
- Strobe Inputs For Receiver Selection
- Gate Inputs For Logic Versatility
- TTL or DTL Drive Capability
- High D-C Noise Margin
- '107A and '107B Have Totem-Pole Outputs
- '108A and '108B Have Open-Collector Outputs
- "B" Versions Have Diode-Protected Input Stage For Power-Off Condition

SN55107A, SN55107B, SN55108A, SN55108B . . . J DUAL-IN-LINE PACKAGE
SN75107A, SN75107B, SN75108A, SN75108B . . . J OR N DUAL-IN-LINE PACKAGE
(TOP VIEW)



NC – No internal connection

FUNCTION TABLE

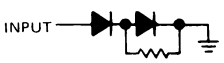
DIFFERENTIAL INPUTS A-B	STROBES G	S	OUTPUT Y
$V_{ID} \geq 25$ mV	X	X	H
-25 mV $< V_{ID} < 25$ mV	X	L	H
	L	X	H
	H	H	INDETERMINATE
$V_{ID} \leq -25$ mV	X	L	H
	L	X	H
	H	H	L

H = high level, L = low level, X = irrelevant

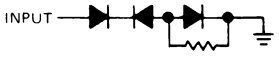
description

These circuits are TTL/DTL compatible high-speed line receivers. Each is a monolithic dual circuit featuring two independent channels. They are designed for general use as well as such specific applications as data comparators and balanced, unbalanced, and party-line transmission systems. These devices are unilaterally interchangeable with and replace SN55107, SN55108, SN75107, and SN75108, but offer diode-clamped strobe inputs to simplify circuit design.

The essential difference between the "A" and "B" versions can be seen in the schematics. Input-protection diodes are in series with the collectors of the differential-input transistors of the "B" versions. These diodes are useful in certain "party-line" systems that may have multiple V_{CC+} power supplies and may be operated with some of the V_{CC+} supplies turned off. In such a system, if a supply is turned off and allowed to go to ground, the equivalent input circuit connected to that supply would be as follows:



"A" VERSION



"B" VERSION

This would be a problem in specific systems that might possibly have the transmission lines biased to some potential greater than 1.4 volts.

The SN55107A, SN55107B, SN55108A, and SN55108B, are characterized for operation over the full military temperature range of -55°C to 125°C. The SN75107A, SN75107B, SN75108A, and SN75108B are characterized for operation from 0°C to 70°C.

FIG. 35. - Piedinatura e tabella funzionale del doppio ricevitore di linea SN 75107 (Texas Instruments).

16. Alimentazione e disaccoppiamenti

L'uso di una attenta progettazione circa la disposizione dei circuiti integrati sulla scheda di circuito stampato, può risultare molto efficiente per ridurre i problemi di accoppiamento e di rumore. Un corretto montaggio non dovrebbe, quindi, prescindere dalle seguenti tre regole:

1) usare una disposizione a griglia per le masse, come è mostrato nella figura 36. Ciò richiede l'uso di schede a doppia faccia, nelle quali le tracce verticali sono su una faccia mentre quelle orizzontali sull'altra.

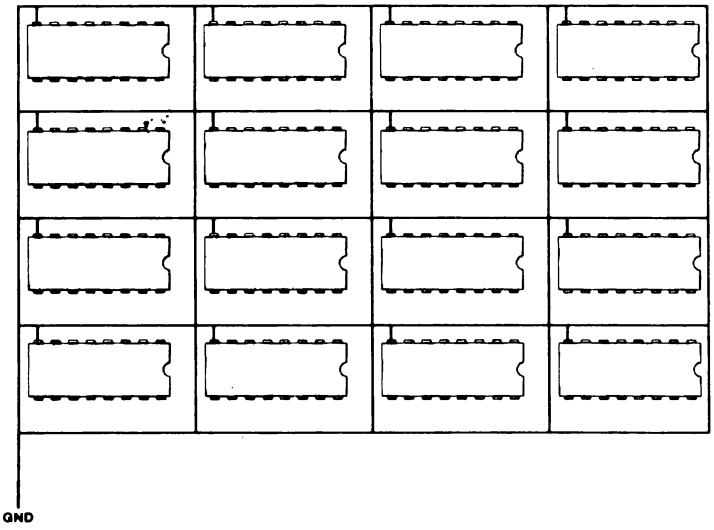


FIG. 36. - Grigliatura per le masse (Intel).

Ad ogni incrocio viene effettuata la loro interconnessione in modo da ridurre l'impedenza caratteristica poiché la connessione di N linee in parallelo, diminuisce l'impedenza di un valore 1/N.

2) usare una disposizione a griglia per l'alimentazione come è mostrato nella figura 37.

3) usare condensatori di disaccoppiamento come è mostrato nelle figure 38. Essi debbono essere posti il più vicino possibile ai piedini di alimentazione e massa fra i quali sono connessi. La loro funzione è di smorzare gli eventuali transienti sulle linee di alimentazione, fornendo una sorgente di energia alle extra correnti richieste ($i = C\ dv/dt$).

Per vedere come ciò avvenga, si analizzi quello che accade quando un congegno richiede extra corrente come ad esempio nel caso del comando RD per una memoria. I condensatori di disaccoppiamento per alte frequenze (normalmente di tipo ceramico e del valore di 0,1 μF), forniscono istantaneamente tale corrente.

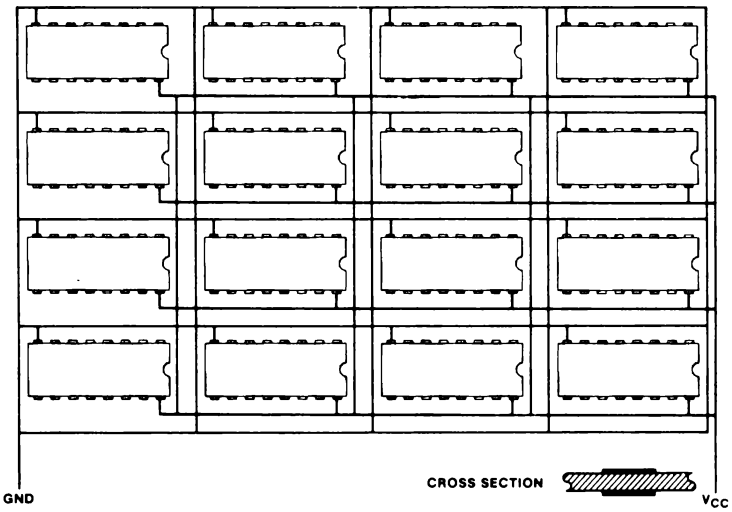


FIG. 37. - Grigliatura per alimentazione e massa (Intel).

I condensatori elettrolitici di più elevato valore, che hanno una risposta in frequenza inferiore e che usualmente vengono disposti all'ingresso della scheda, riforniscono allora di cariche i precedenti, tramite la bassa impedenza della distribuzione a griglia. Infine, l'alimentatore, che ha una risposta in frequenza molto bassa, ricarica i condensatori elettrolitici e si ritorna all'equilibrio.

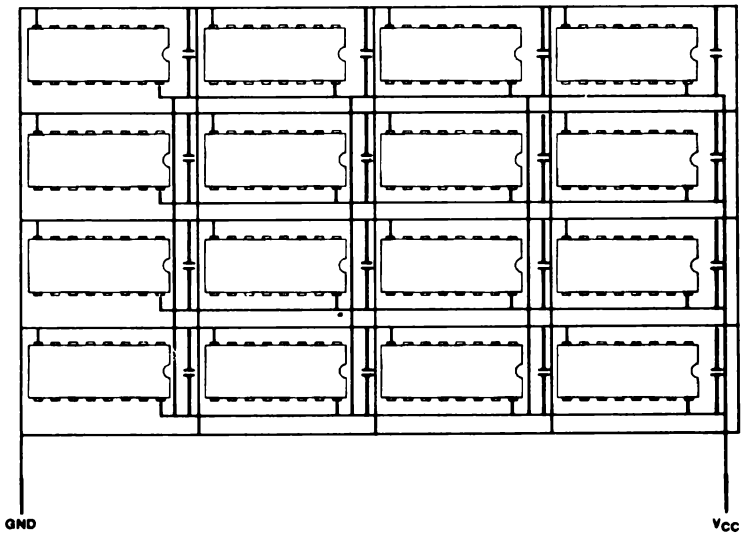


FIG. 38. - Posizione dei condensatori di disaccoppiamento (schema elettrico).

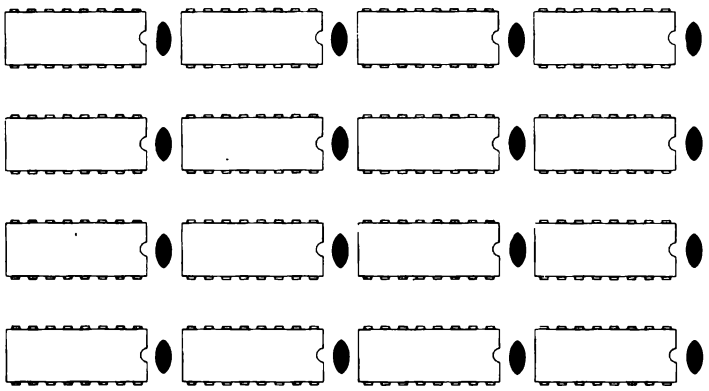


FIG. 38bis. - Posizione dei condensatori di disaccoppiamento (Intel).

17. I bus per microcomputer

Una delle prime decisioni che il progetto di un sistema a microprocessore impone, è la scelta del tipo di bus da utilizzarsi per l'interconnessione delle singole schede di un sistema modulare.

Dalla scelta del *bus del sistema* sono influenzate le prestazioni, la flessibilità ed il tipo di applicazioni consentite per cui ne diventa una parte fondamentale.

La utilizzazione di un bus standardizzato oppure di uno creato appositamente dal progettista, non ha posto particolari problemi fino a quando si utilizzava un solo microprocessore con componenti della stessa famiglia, ed i periferici erano poco sofisticati.

Infatti, per i primi non veniva a crearsi alcuna incompatibilità di bus, mentre per i secondi le linee di controllo erano in numero estremamente limitato.

Nel passato, vari comitati e diverse case costruttrici hanno cercato di progettare bus standard che definissero i criteri e gli schemi per le interconnessioni elettriche, meccaniche e funzionali.

Però ciascuno di essi assommava vantaggi e svantaggi per cui nessuno è riuscito ad affermarsi sopra ad ogni altro.

Per comprendere la natura dei problemi incontrati nella realizzazione di un bus standardizzato si può procedere, come esempio, alla definizione delle prestazioni richiestegli per l'implementazione di una architettura multiprocessore. Un sistema a più microprocessori ha per fine il miglioramento delle prestazioni mediante la suddivisione delle funzioni, per cui il bus del sistema deve porre particolare importanza sui seguenti requisiti:

	largh. dati	largh. indirizzi	trasmissione	numero di pin	capacità di multiproc.
VERSAbus	8/16/32	32	asincrona non mux	120/140	si
MULTIBUS IEEE 796	8/16	24	asincrona non mux	60/ 86	si
UNIBUS	8/16	8/16	asincrona non mux	56/ 72	no
LSI - 11	8/16	16	asincrona mux	56/ 72	no
S-100 IEEE 696	8/16	16	asincrona non mux	100	no
MUBUS EUROBUS	16	16	asincrona non mux	51/ 74	si
STD bus	8	16	sincrona non mux	56	no

Tabella 2. - Caratteristiche di alcuni fra i più noti bus.

a) linee di trasmissione con caratteristiche elettriche e temporali adatte anche per elevate velocità. Adozione di connettori standard e fattori di forma delle schede a circuito stampato (PC) adatti ad espansioni modulari.

b) mantenere separate le funzioni svolte da ciascun microprocessore, in modo da evitare mutue interferenze.

c) sincronizzare le varie attività dei diversi μP.

d) permettere un rapido accesso alle diverse risorse del sistema.

e) utilizzare protocolli standard di comunicazione seriale o parallela.

Nella tabella 2 sono riassunte alcune caratteristiche scelte tra i bus più noti sul mercato. Nel seguito ne viene data una succinta descrizione, demandando ai rispettivi manuali una più appropriata trattazione.

VERSAbus

È stato introdotto dalla Motorola ed è compatibile con μP da 8 a 32 bit. Permette l'implementazione, in modo economico, di architetture multiprocessore dedicate a controlli industriali e comunicazioni. Tutti i segnali, in numero molto elevato, sono distribuiti su due connettori a circuito stampato (P_1 e P_2) in modo di avere la possibilità di utilizzarne solo uno per sistemi di ridotte dimensioni.

Nella figura 39 è mostrato lo standard fisico per le schede.

Al connettore P_1, di 140 piedini, fanno capo: dati, indirizzi, controlli asincroni, priorità per le interruzioni, livelli di arbitraggio del bus, alimentazioni, controlli.

Al connettore P_2 di 120 piedini, fanno capo: linee di controllo di I/O, linee di trasmissione seriali, alimentazioni, espansioni per dati ed indirizzi.

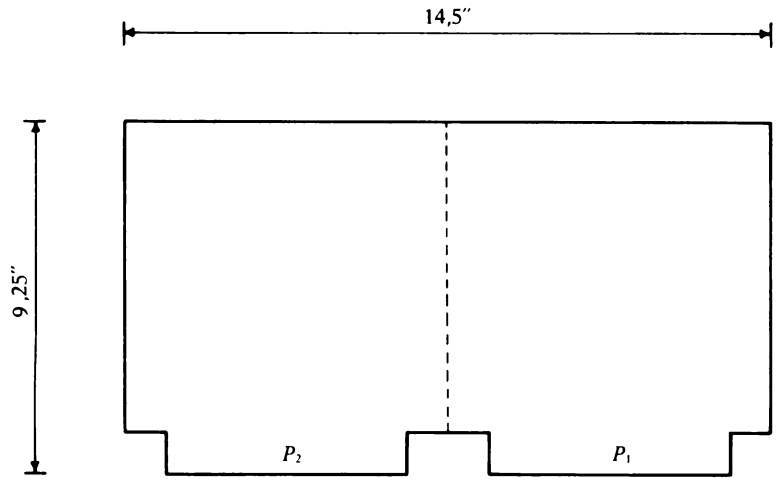


FIG. 39. - Standard fisico per l'interfaccia VERSAbus.

MULTIBUS

Questo tipo di bus standard fu inizialmente realizzato dalla Intel per le sue piastre SBC, ma successivamente è stato adottato anche a livello internazionale con la sigla IEEE-796 data la sua struttura estremamente generalizzata. Per un maggior approfondimento circa le specifiche elettriche, temporali e meccaniche, è bene far riferimento al manuale «Intel multibus specification» ma è comunque utile riassumerne gli aspetti più importanti.

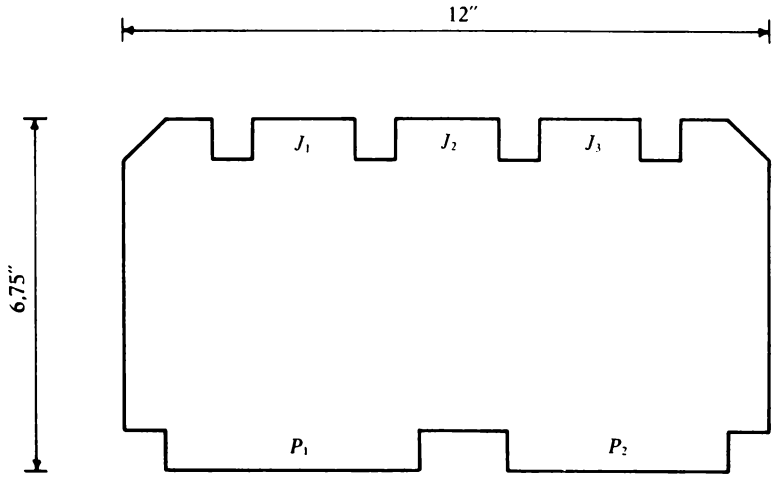


FIG. 40. - Connessioni del multibus.

Nella figura 40 è mostrata la scheda a circuito stampato di un modulo master contenente un computer completo.

I tre connettori all'estremità superiore, indicati con J_1, J_2 e J_3, sono opzionali ed utilizzati per le comunicazioni parallelo o seriale con dispositivi esterni. La comunicazione con gli altri moduli del sistema avviene invece, tramite gli 86 piedini del connettore P_1 ed i 60 del connettore P_2.

I segnali di P_1 sono suddivisi nelle seguenti categorie:
- indirizzi
- dati
- controlli
- interruzioni
- gestione del bus
- alimentazione

come è mostrato nella tabella 3.

I segnali relativi al connettore P_2 non sono previsti nel collegamento a bus e sono utilizzati per applicazioni particolari.

MUBUS/EUROBUS

Questa interfaccia è nata dalla collaborazione tra i politecnici di Torino, e Losanna.

Lo standard fisico consiste di schede di formato Eurocard con 74 contatti (37 + 37) con passo di 2,54 mm; la standardizzazione funzionale è invece illustrata in figura 41. Si può osservare che sono presenti:
- 8 linee per le varie alimentazioni;
- una linea per segnale di rete TTL compatibile;
- 16 linee per gli indirizzi;
- 16 linee per i dati;
- 18 linee per segnali di comando e controllo.

La struttura offre le seguenti possibilità:

a) gestione distribuita della priorità del servizio dei periferici mediante la tecnica «daisy chain» per le richieste di interruzione;

b) concatenazione della richiesta di HOLD;

c) utilizzazione di più di una CPU (la linea INB è riservata alla richiesta del bus da parte di una seconda CPU);

d) rinfresco delle memorie dinamiche per microprocessori che hanno questa caratteristica.

L'impiego del MUBUS è particolarmente indicato per scopi didattici e di ricerca, poiché la sua elevata flessibilità consente la composizione di strutture di varia mole e complessità secondo le esigenze del singolo utente.

		(COMPONENT SIDE)			(CIRCUIT SIDE)	
	PIN	MNEMONIC	DESCRIPTION	PIN	MNEMONIC	DESCRIPTION
POWER SUPPLIES	1	GND	Signal GND	2	GND	Sig GND
	3	+5V	+5Vdc	4	+5V	+5Vdc
	5	+5V	+5Vdc	6	+5V	+5Vdc
	7	+12V	+12Vdc	8	+12V	+12Vdc
	9	-5V	-5Vdc	10	-5V	-5Vdc
	11	GND	Signal GND	12	GND	Signal GND
BUS CONTROLS	13	BCLK/	Bus Clock	14	INIT/	Initialize
	15	BPRN/	Bus Pri.In	16	BPRO/	Bus Pri.Out
	17	BUSY/	Bus Busy	18	BREQ/	Bus Request
	19	MRDC/	Mem Read Cmd	20	MWTC/	Mem Write Cmd
	21	IORC/	I/O Read Cmd	22	IOWC/	I/O Write Cmd
	23	XACK/	XFER Acknowiedge	24	INH1/	Inhibit 1 disable RAM
BUS CONTROLS AND ADDRESS	25		Reserved	26	INH2/	Inhibit 2 disable
	27	BHEN/	Byte High Enable	28	AD10/	PROM OR ROM
	29	CBRQ/	Common Bus Request	30	AD11/	Address
	31	CCLK/	Constant Clk	32	AD12/	Bus
	33	INTA/	Intr Acknowledge	34	AD13/	
INTERRUPTS	35	INT6/	Parallel	36	INT7/	Parallel
	37	INT4/	Interrupt	38	INT5/	Interruput
	39	INT2/	Requests	40	INT3/	Requests
	41	INTO/		42	INTO/	
ADDRESS	43	ADRE/		44	ADRF/	
	45	ADRC/		46	ADRD/	
	47	ADRA/	ADDRESS	48	ADRB	Address
	49	ADR8/	Bus	50	ADR9/	Bus
	51	ADR6/		52	ADR7/	
	53	ADR4/		54	ADR5/	
	55	ADR2/		56	ADR3/	
	57	ADRO/		58	ADR1/	
DATA	59	DATE/		60	DATF/	
	61	DATC/		62	DATD/	
	63	DATA/	Data	64	DATB/	data
	65	DAT8/	Bus	66	DAT9/	Bus
	67	DAT6/		68	DAT7/	
	69	DAT4/		70	DAT5/	
	71	DAT2/		72	DAT3/	
	73	DATO/		74	DAT1/	
POWER SUPPLIES	75	GND	Signal GND	76	GND	Signal GND
	77		Reserved	78		Reserved
	79	-12V	-12Vdc	80	-12V	-12Vdc
	81	+5V	+5Vdc	82	+5V	+5Vdc
	83	+5V	+5Vdc	84	+5V	+5Vdc
	85	GND	Signal GND	86	GND	Signal GND

Tabella 3. - Segnali per il connettore P_1 dell'interfaccia Multibus.

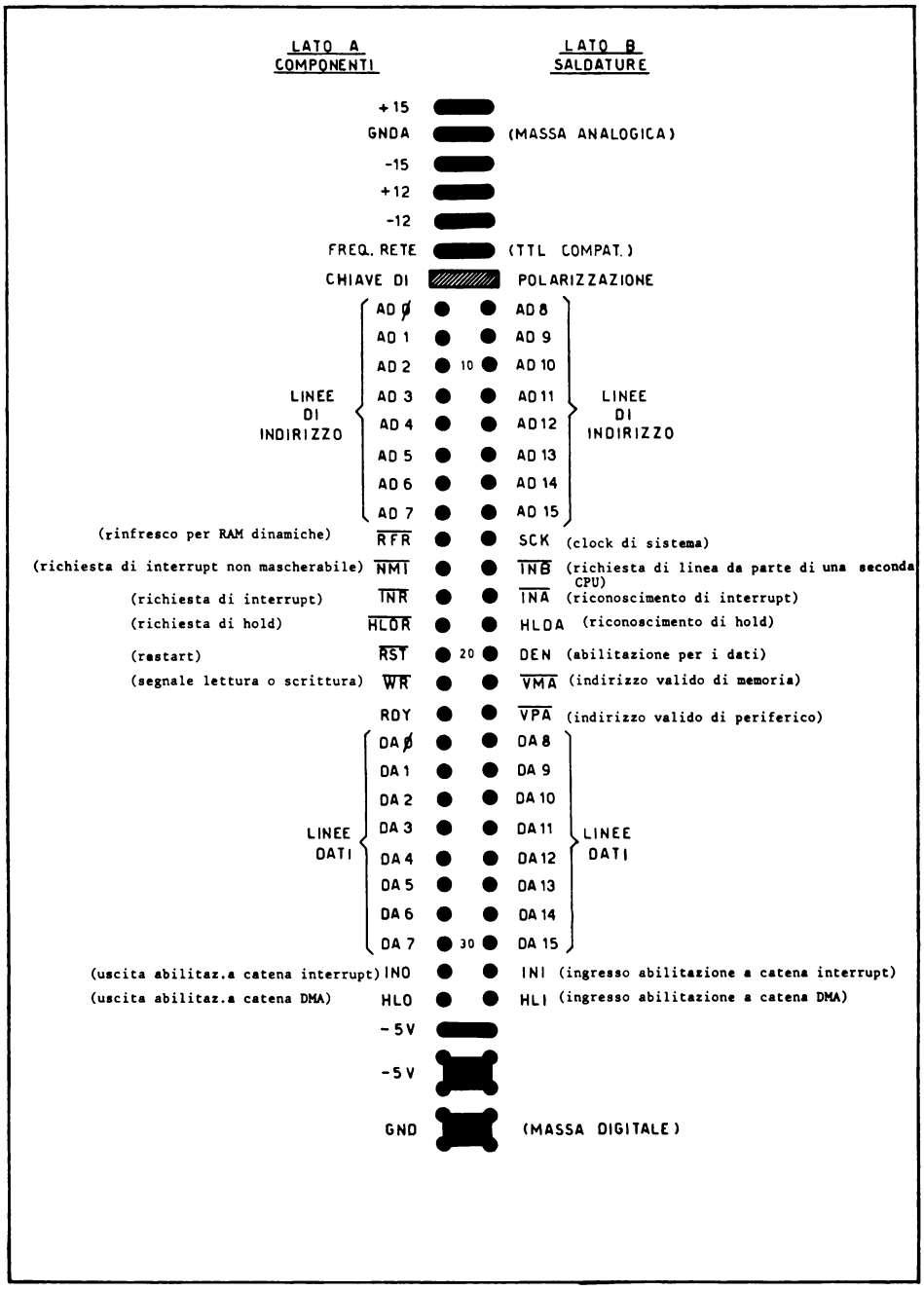


FIG. 41. - Standard fisico e funzionale per l'interfaccia MUBUS.

UNIBUS ed LSI-11

Entrambi questi bus sono stati introdotti ed impiegati dalla Digital Equipment e dai costruttori che producono piastre compatibili. Il primo supporta i minicomputer della famiglia PDP-11, mentre il secondo è utilizzato per l'interconnessione delle schede della famiglia LSI-11.

S-100 bus

Il suo nome deriva dal numero di linee utilizzate (100), suddivise in due

LATO COMPONENTI

pin	funzione	mnemonico	descrizione
1	alimentazione	+ 5 V	+ 5 V DC
3		GROUND	massa digitale
5		- 5 V	- 5V DC
7	BUS	D3	dati di ordine basso
9	DATI	D2	dati di ordine basso
11		D1	dati di ordine basso
13		D0	dati di ordine basso
15		A7	indirizzi di ordine basso
17		A6	indirizzi di ordine basso
19	BUS	A5	indirizzi di ordine basso
21		A4	indirizzi di ordine basso
23	INDIRIZZI	A3	indirizzi di ordine basso
25		A2	indirizzi di ordine basso
27		A1	indirizzi di ordine basso
29		A0	indirizzi di ordine basso
31		$\overline{WR}$	scrittura
33		$\overline{IORQ}$	selezione I/O
35	BUS	$\overline{IOEXP}$	espansione I/O
37		$\overline{REFRESH}$	temporizzazione di refresh
39	DEI	$\overline{STATUS}$ 1	stato della CPU
41		$\overline{BUSAK}$	riconoscimento del bus
43	CONTROLLI	$\overline{INTAK}$	riconoscimento interruzione
45		$\overline{WAITRQ}$	richiesta di wait
47		$\overline{SYSRESET}$	reset del sistema
49		$\overline{CLOCK}$	clock
51		PCO	uscita Priority Chain
53	BUS.	AUX GND	altre masse
55	ALIMENTAZ.	AUX + V	altre alimentaz. (+ 12 V)

Tabella 4. - Segnali del bus STD-Z80 per il lato componenti.

gruppi di 50 per ciascuna faccia della scheda a circuito stampato. Fu inizialmente introdotto per il mercato dei «Personal computer» utilizzanti come CPU l'8080 della Intel.

Successivamente è stata ampliata e standardizzata come IEEE-696, rendendola compatibile per microprocessori sia ad 8 che a 16 bit ed in grado di realizzare il colloquio multiprocessore.

LATO CIRCUITO STAMPATO

pin	funzione	mnemonico	descrizione
2	alimentazione	+ 5 V	+ 5 V DC
4		GROUND	massa digitale
6		- 5V	- 5 V DC
8		D7	dati di ordine alto
10	BUS	D6	dati di ordine alto
12		D5	dati di ordine alto
14	DATI	D4	dati di ordine alto
16		A15	indirizzi di ordine alto
18		A14	indirizzi di ordine alto
20	BUS	A13	indirizzi di ordine alto
22		A12	indirizzi di ordine alto
24	INDIRIZZI	A11	indirizzi di ordine alto
26		A10	indirizzi di ordine alto
28		A9	indirizzi di ordine alto
30		A8	indirizzi di ordine alto
32		$\overline{RD}$	lettura
34	BUS	$\overline{MEMRQ}$	selezione memoria
36		$\overline{MEMEX}$	espansione memoria
38		$\overline{MCSYNC}$	sincronizzazione CPU
40	DEI	$\overline{STATUS}$ 0	stato della CPU
42		$\overline{BUSRQ}$	richiesta del bus
44	CONTROLLI	$\overline{INTRQ}$	richiesta di interruzione
46		$\overline{NMIRQ}$	interuzz. non mascherabile
48		$\overline{PBRESET}$	pulsante di reset
50		$\overline{CNTRL}$	altre temporizzazioni
52		PCI	ingresso Priority Chain
54	BUS	AUX GND	altre masse
56	ALIMENTAZ.	AUX - V	altre alimentaz. (-12 V)

Tabella 5. - Segnali del bus STD-Z80 per il lato circuito.

STD bus

È stata realizzata congiuntamente dalla Mostek e Pro-Log per permettere di utilizzare piastre di piccole dimensioni ed a basso costo nel campo dei controlli industriali supportati da microprocessori ad otto bit. Nelle tabelle 4 e 5 sono mostrati i segnali relativi al bus STD-Z80 relativo a tale CPU.

18. Trasmissione delle informazioni mediante fibre ottiche

La trasmissione delle informazioni per via ottica offre notevoli vantaggi in parecchi campi, quali ad esempio:
— controlli industriali, sistemi di misura e di regolazione, computer, apparecchiature medicali.
— tecnica delle misure per alta tensione, centrali elettriche.
— trasmissione d'informazioni non intercettabili in campo civile e militare
— zone con pericolo di esplosioni nelle raffinerie e nell'industria chimica e mineraria, zone con atmosfera corrosiva
— reti di bordo di aerei e navi
— sistemi di trasmissione a larga banda

Le caratteristiche particolari di questa trasmissione sono:
— insensibilità ai disturbi elettromagnetici
— isolamento elettrico tra trasmettitore e ricevitore
— nessuna irradiazione di segnale e quindi nessun problema di diafonia
— assenza di scintille tra i terminali di giunzione
— elevata flessibilità con piccole dimensioni e peso ridotto
— attenuazione indipendente dalla frequenza e quindi possibilità di trasmissioni a larga banda.

Componenti di una linea di trasmissione ottica

Una linea per la trasmissione ottica delle informazioni è costituita essenzialmente da un trasmettitore ottico, da un cavo a fibre ottiche e da un ricevitore ottico.

Il trasmettitore (si tratta di un trasduttore elettro-ottico) converte il segnale elettrico preventivamente amplificato in un segnale luminoso (fig. 41) che attraversa il cavo propagandosi per riflessione totale o rifrazione (fibre con indice di riflessione a profilo graduale) ed arriva ad un ricevitore (trasduttore ottico-elettrico) che lo riconverte in segnale elettrico.

Trasduttore elettro-ottico

I diodi luminosi (LED) e i diodi laser sono idonei per la trasmissione ottica delle informazioni, hanno quasi le stesse dimensioni delle fibre e possono essere modulati in modo relativamente facile.

La loro lunghezza d'onda d'esercizio va da 800 a 900 nm, con fibre a bassa attenuazione. Mentre i diodi laser, grazie alle loro particolari caratteristiche, vengono impiegati in prevalenza in telefonia, i LED consentono invece di realizzare linee di trasmissione affidabili ed economiche per piccole e medie distanze, fino ad alcuni chilometri senza ripetitori intermedi.

Trasduttore ottico-elettrico

Il compito del ricevitore ottico consiste nel convertire in potenza elettrica l'intensità luminosa all'uscita del cavo. I diodi PIN, impiegati a tale scopo, hanno una elevata sensibilità per captare la lunghezza d'onda dei segnali luminosi emessi dai diodi trasmittenti.

Una ulteriore particolarità di questi rivelatori a semiconduttori è quella di avere brevi tempi di salita.

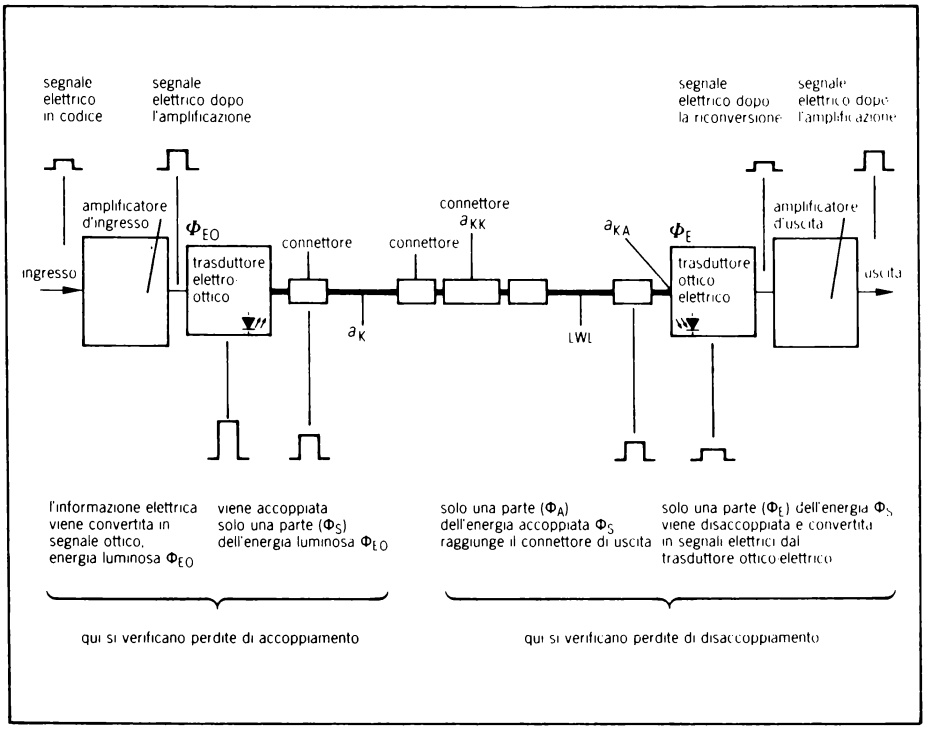


FIG. 41. - Trasmissione delle informazioni per via ottica (SIEMENS)

Parametri ottici ed elettrici

Φ_{EO} (μW) energia luminosa generata dal trasduttore elettro-ottico
Φ_s (μW) energia luminosa accoppiata al cavo e fibre ottiche
Φ_A (μW) energia luminosa disaccoppiata dal cavo a fibre ottiche
Φ_E (μW) energia luminosa incidente sulla superfice attiva del ricevitore a_K
a_K(dB) attenuazione specifica del cavo a fibre ottiche
a_{KK} (dB) attenuazione di transito di un connettore
a_{KA} (dB) perdite di accoppiamento tra connettore e trasduttore ottico-elettrico.
l (km) lunghezza del cavo a fibre ottiche.

L'attenuazione dell'intero sistema è:

$$l \cdot a_K + a_{KK} + a_{KA} = 10 \log \frac{\Phi_s}{\Phi_e} \text{ (dB)}$$

La tabella 5 consente di calcolare il rapporto di energia e quindi l'energia luminosa disaccoppiata

La corrente fotoelettrica all'uscita del trasduttore ottico-elettrico è:
I_0 (μA) $= S \cdot \Phi_E$

La sensibilità S del trasduttore ottico-elettrico dipende dalla lunghezza d'onda dell'energia luminosa.

dB	$n = \Phi_S/\Phi_E$	dB	$n = \Phi_S/\Phi_E$
0	1	10	10
1	1,26	20	100
2	1,6	30	1000
3	2,0	40	10^4
4	2,5	50	10^5
5	3,2	60	10^6
6	4,0	70	10^7
7	5,0	80	10^8
8	6,3	90	10^9
9	7,9	100	10^{10}

Tabella 5.

La lunghezza massima della linea con velocità di trasmissione prestabilita, dipende essenzialmente dai parametri del trasmettitore, del cavo e del ricevitore. L'energia luminosa Φ_S, accoppiata all'ingresso del cavo, è stabilita dalla corrente del trasduttore elettro-ottico. L'intensità luminosa minima Φ_E, necessaria per il ricevitore ottico (rivelatore), è determinata in base al rapporto segnale/rumore richiesto o al tasso d'errore bit consentito.

Per ricevitori semplici ed economici viene scelto di solito il valore $\Phi_E = 0{,}5/\mu W$.

L'attenuazione massima non superabile è quindi uguale a:

$$D = 10 \log \frac{\Phi_S}{\Phi_E} \text{ (dB)}$$

Esempio:
per $\Phi_S = 30\,\mu W$ con $I_F = 100$ mA e $\Phi_E = 0{,}5\,\mu W$ si ottiene D = 17,5 dB

Oltre ai valori di attenuazione già rilevati a_K, a_{KK}, e a_{KA} bisogna tener conto delle seguenti perdite:
a_t caduta dell'emissione luminosa in funzione della durata d'esercizio, tipica $\leqq$ 1 dB e
a_{TK} diminuzione dell'intensità luminosa nel campo della temperatura di la-

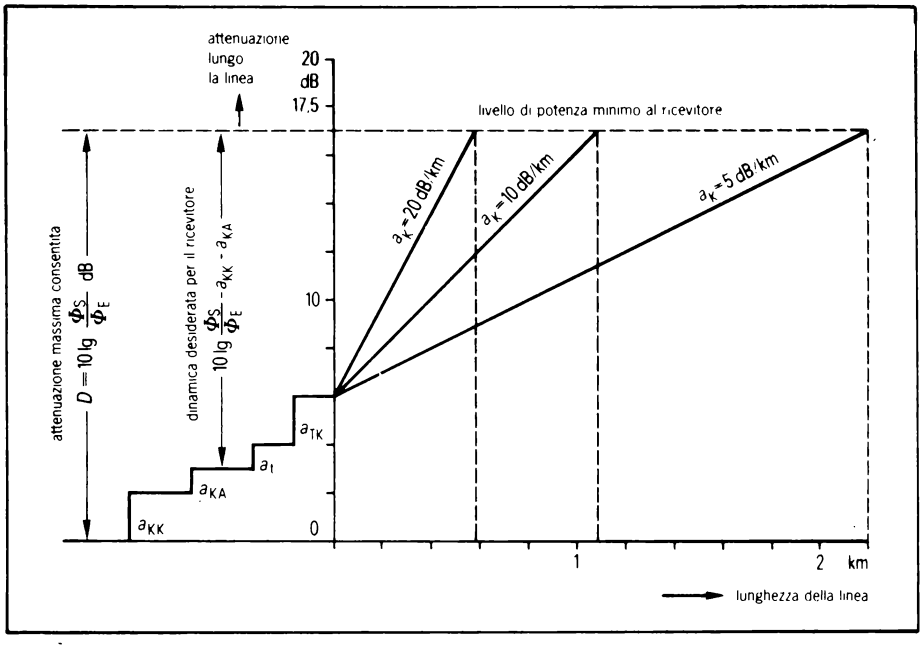


FIG. 42. - Diagramma della intensità luminosa di una linea di trasmissioen ottica (SIEMENS).

voro (coefficiente di temperatura dell'intensità luminosa TK X aumento della temperatura Δ T) in dB.

Dal coefficiente D e dalle perdite totali si ricava la lunghezza della linea l.

$l = 10 \log \Phi_S / \Phi_E - a_{KK} - a_{KA} - a_t - a_{TK}$

Nel diagramma di fig. 42 sono riportate le lunghezze massime della linea in funzione di diversi valori di a_K. Dal diagramma è possibile rilevare inoltre la dinamica del ricevitore $DY = 10 \log \Phi_S / \Phi_E - a_{KK} - a_K$. Il valore della dinamica è sufficiente solo quando il ricevitore può essere attivato, senza saturarsi sia dalle linee di lunghezza massima, sia da quelle più brevi. Una sovramodulazione determinerebbe un aumento del fattore di distorsione del tasso di errore dei bit.

Tempi di salita e propagazione degli impulsi lungo una linea di trasmissione ottica

Vengono presi in considerazione i tempi di salita riportati nei dati tecnici dei trasduttori e l'allargamento dell'impulso determinato dal cavo ottico. In questo caso si suppone che il tempo di salita dell'impulso all'ingresso del trasduttore elettro-ottico sia $t_E = 0$

$t_{E0(ns)}$ tempo di salita del trasduttore elettro-ottico (trasmettitore)
$t_{0E(ns)}$ tempo di salita del trasduttore ottico-elettrico (ricevitore)
t_d (ns/km) allargamento dell'impulso nel cavo ottico.
l (km) lunghezza del cavo ottico.

I tempi di salita del trasmettitore e del ricevitore e l'allargamento dell'impulso del cavo ottico influiscono sulla frequenza limite e quindi sulla capacità di trasmissione

Quando si calcolano i tempi è necessario considerare sia le caratteristiche dinamiche dei componenti sia i tempi di transito del segnale (ritardi degli impulsi) lungo la linea.

Il tempo di salita totale è

$$t_r = \sqrt{t_{E0}^2 + t_{0E}^2}$$

La larghezza dell'impulso all'uscita del trasduttore ottico-elettrico è: $T = T_0 + T_D = T_0 + l \cdot t_p$ dove T_0 è la larghezza dell'impulso all'ingresso del trasduttore elettro-ottico e t_p il tempo di propagazione.

Cavi ottici

In realtà una fibra ottica non è costituita da un solo materiale omogeneo, ma da diversi strati concentrici di materiale con diverse caratteristiche. Al centro vi è un cilindro di materiale trasparente che viene chiamato core (nucleo), attorno ad esso vi è un tubo coassiale costituito anch'esso da materiale trasparente ma con diverso indice di rifrazione rispetto al core.

Questo secondo strato detto cladding (mantello) completa la struttura della fibra ottica dal punto di vista del funzionamento come guida d'onda dielettrica. Un ulteriore strato di materiale plastico, poi, ha la funzione di rendere meccanicamente più resistente la fibra. Occorre notare che il termine fibra ottica è molto generico in quanto comprende tutta una gamma di fibre realizzate con materiali molto diversi quali il vetro, la plastica etc.

Nel caso di una trasmissione bidirezionale delle informazioni naturalmente sono necessarie due fibre ottiche, ognuna per ogni senso di trasmissione: l'utilizzo di una sola fibra ottica porterebbe all'uso di due portanti luminose a diversa lunghezza d'onda, soluzione quest'ultima attualmente non economica.

I cavi ottici vengono impiegati per trasmettere segnali ottici e possono essere formati da una sola fibra o da parecchie fibre raggruppate a fascio. A seconda della struttura della fibra e del relativo sistema di propagazione della luce, si hanno fibre ad indice di rifrazione con profilo a gradino o con profilo graduale. Le prime sono costituite da un nucleo e da un rivestimento con indici di rifrazione diversi, per cui la luce si propaga per riflessione totale lungo il confine nucleo-rivestimento. Le seconde fibre presentano un indice di rifrazione con profilo inclinato parabolicamente dal centro del nucleo verso l'esterno, per cui la luce si propaga dal centro alla periferia con basso indice di rifrazione rispetto all'asse della fibra, questo tipo di fibra consente quindi elevate velocità di trasmissione.

Collegamento di sistemi digitali per via ottica

I trasduttori elettro-ottici, analogamente a quanto avviene per gli accoppiatori ottici (vol. Elettronica Digitale), possono essere interfacciati direttamente con sistemi TTL standard. Un semplice collegamento mediante fibre ottiche tra due sistemi digitali è quello mostrato in fig. 43.

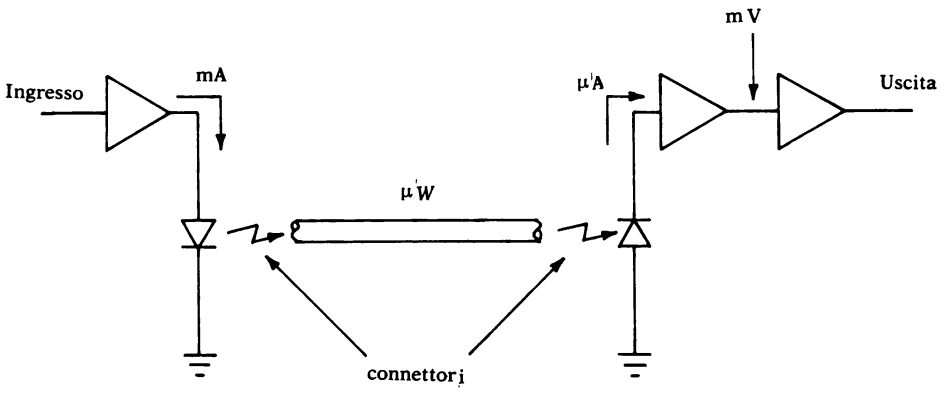


FIG. 43. - Trasmissione digitale delle informazioni per via ottica.

Il segnale digitale viene prima bufferizzato e quindi convertito in un segnale luminoso tramite il diodo LED. La trasmissione dell'informazione avviene mediante un cavo a fibre ottiche collegato sia alla parte trasmittente che a quella ricevente tramite un connettore. Il segnale trasmesso viene poi riconvertito in segnale elettrico (corrente dell'ordine dei μA) e quindi amplificato tramite uno o più stadi amplificatori.

Di solito nei sistemi digitali sono utilizzati prevalentemente dei diodi LED come sorgenti di luce per il loro basso costo associato ad una lunga durata. Per velocità di trasmissione maggiori occorre invece far uso di laser.

In fig. 44 è mostrato lo schema di un trasmettitore ottico interfacciato con una logica TTL mediante il buffer Nand SN 7438 (uscita open collector, V_{0L} = 0,4 V ed I_{OLMax} = 58mA).

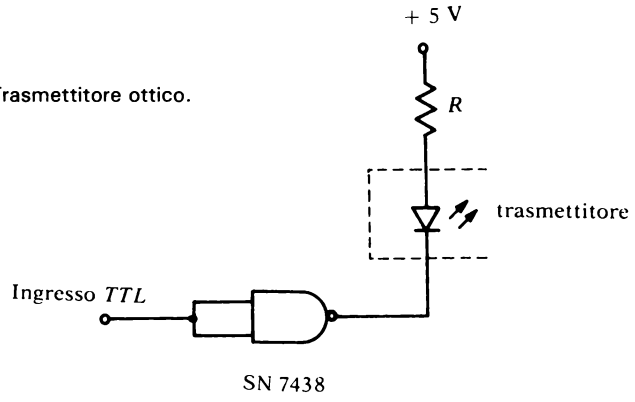


FIG. 44. - Interfaccia TTL - Trasmettitore ottico.

Il valore della resistenza deve essere fissato in modo da avere al massimo una corrente I_F uguale alla I_{0LMax} consentita alla porta logica:

$$R_{Min} = \frac{V_{CC} - V_F - V_{0L}}{I_{0LMax}} = 52\Omega$$

tenendo presente che la tensione V_F ai capi del diodo LED in conduzione si aggira intorno a 1,6 V.

Per la trasmissione delle informazioni a più lunghe distanze possono essere usati i circuiti di fig. 45 e 46 in cui il trasmettitore ottico è interfacciato con un SN 75450 (Texas), driver capace di fornire corrente fino a un massimo di 300 mA.

Nel caso del circuito di fig. 45 la I_F è data da:

$$I_F = \frac{V_{CC} - V_{CESat} - V_F}{R}$$

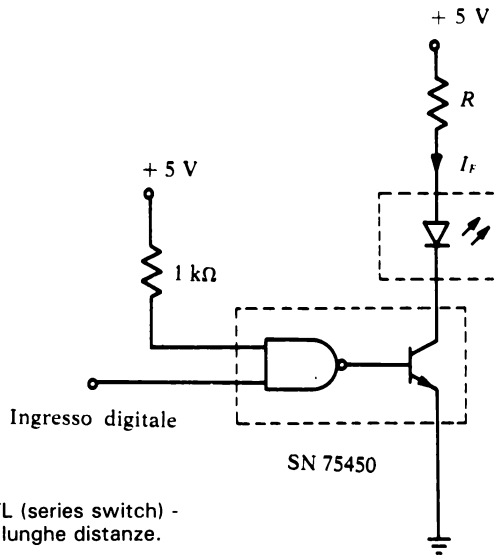


FIG. 45. - Interfaccia TTL (series switch) - Trasmettitore ottico per lunghe distanze.

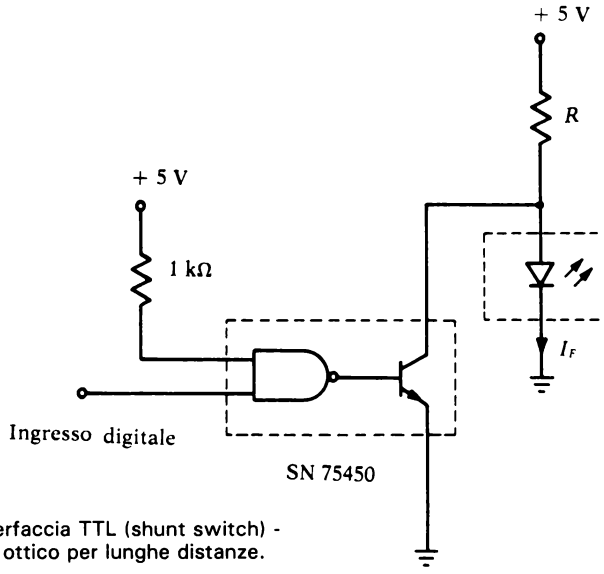


FIG. 46. - Interfaccia TTL (shunt switch) - Trasmettitore ottico per lunghe distanze.

Il controllo mediante shunt switch (fig. 46) anziché interrompere bruscamente la corrente come in quello di tipo serie, permette di mantenere la corrente quasi costante facendola solamente deviare o nel diodo LED o nel transistore in saturazione.

In questo modo, l'alimentazione non è più costretta a sopportare bruschi sbalzi di corrente, rendendo così inutile l'adozione di condensatori di livellamento posti tra l'alimentazione e massa (parag. 16).

Per velocità di tramissione molto elevate si può far uso di una logica non saturante quale la ECL. In fig. 47 è riportata la parte trasmittente di un circuito operante con questo tipo di logica. Il driver commuta una corrente di 50 mA con tempi di salita e discesa inferiori ai 5 ns. Il LED in questo circuito, un Motorola MF0E103 F0AC, invia alla fibra una potenza ottica di circa 35 μW, centrata attorno ad una lunghezza d'onda di 910 nm, con un tempo di commutazione pari a circa 20 ns.

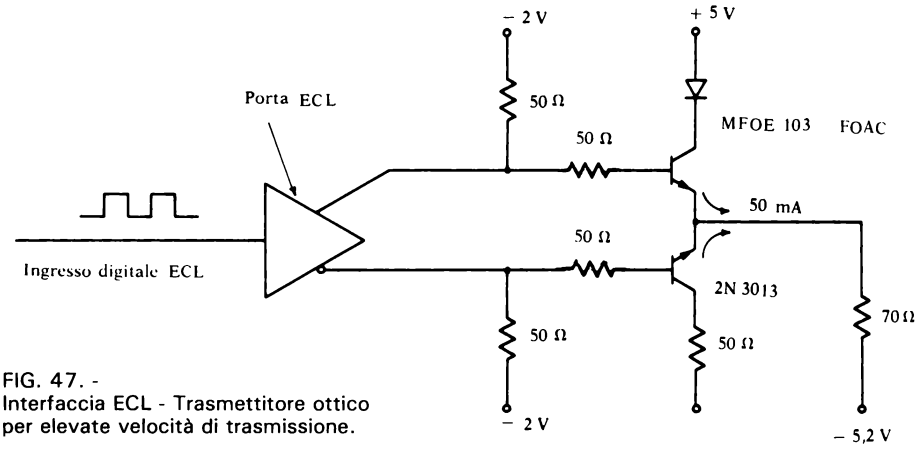


FIG. 47. -
Interfaccia ECL - Trasmettitore ottico per elevate velocità di trasmissione.

In un ricevitore ottico, il trasduttore ottico-elettrico deve essere collegato ad un circuito adatto in modo che il segnale rivelato possa essere com-

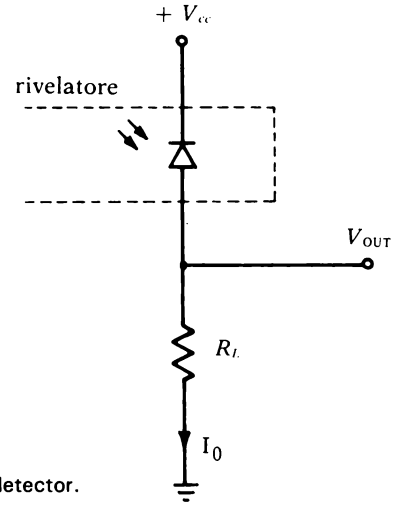


FIG. 48. - Interfacciamento del detector.

patibile con la logica esterna del sistema. Ciò potrebbe essere fatto abbastanza semplicemente servendosi del circuito di fig. 48 in cui la corrente del fotodiodo viene convertita in tensione per mezzo della resistenza R_L, con R_L resistenza di carico del rivelatore (detector)

Infatti la tensione di uscita in questo caso vale:

$$V_0 = I_0 \cdot R_L$$

La soluzione appena vista però non è conveniente in quanto limita considerevolmente la velocità di tramissione essendo legata al valore di R_L poiché il tempo di salita (rise time) del circuito vale:

$t_C = 2{,}2\, R_L \cdot C$

con C capacità di giunzione del fotodiodo. Ne segue che il valore di R_L deve essere così basso in modo che il tempo totale di salita t_r, dovuto all'interazione con la capacità di giunzione del fotodiodo e con quella di carico, risulti piuttosto contenuto.

Il tempo totale di salita infatti è dato dalla relazione:

$$t_r = \sqrt{t_{0E}^2 + t_C^2}$$

Un circuito che elimina le limitazioni dovute al tempo di salita è invece quello di fig. 49 in cui si fa uso di un amplificatore operazionale reazionato.

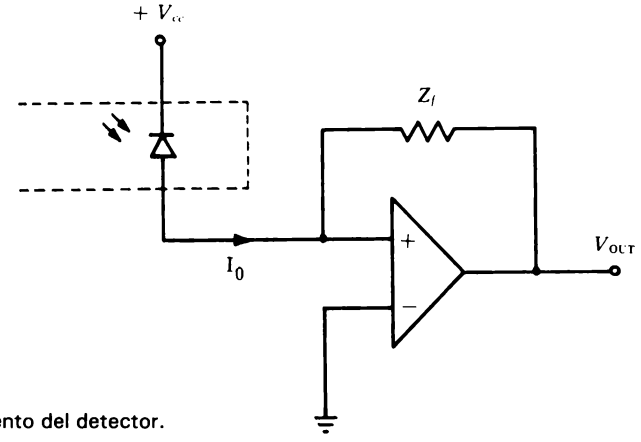


FIG. 49. -
Interfacciamento del detector.

La tensione di uscita, V_0, è ancora data dalla stessa equazione vista precedentemente eccetto che in questo caso la R_L diventa Z_f con Z_f impedenza di reazione dell'amplificatore.

Il carico effettivo del fotodiodo così come l'impedenza di uscita del circuito hanno quindi un valore estremamente basso, ciò permette il pilotag-

gio di capacità di valore relativamente alto.

Il circuito di fig. 49 richiede inoltre il più delle volte l'inserimento di un ulteriore stadio amplificatore in modo da raggiungere i livelli di tensione necessari per il funzionamento del sistema digitale ad esso collegato.

In fig. 50 è mostrato un circuito ricevente compatibile con una trasmissione digitale od analogica.

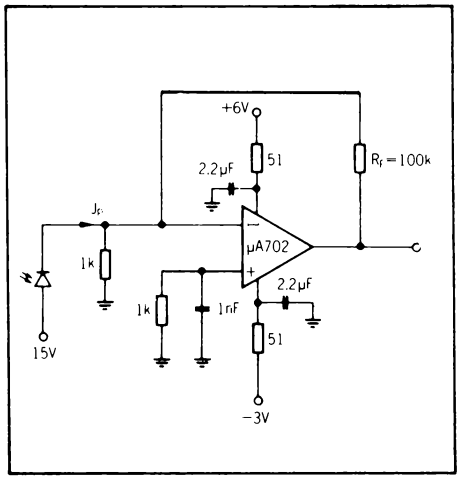


FIG. 50.-
Trasduttore ottico con amplificatore d'uscita a larga banda (SIEMENS).

La tensione di uscita, proporzionale all'intensità luminosa accoppiata, vale:

$$V_0 = I_0 \cdot R_f$$

con I_0 corrente fotoelettrica presente all'uscita del trasduttore. L'amplificazione del circuito può essere variata agendo sul valore di R_f. L'uscita fornisce tensioni fino a 1,5 V_{ss} con tempo di salita di circa 30 ns. Il tempo di salita dell'impulso aumenta però quando la corrente fotoelettrica I_0 (linee brevi con bassa attenuazione e quindi elevata intensità luminosa disaccoppiata Φ_A all'uscita del cavo ottico) e la tensione di uscita sono elevate (oltre 1,5 V_{ss}).

Componenti digitali speciali per la trasmissione ottica delle informazioni

Attualmente sono disponibili sul mercato, in diverse esecuzioni in base ai dati ottici ed elettrici, trasmettitori e ricevitori montati rispettivamente in una piccola custodia compatta.

In questo modo sussiste la possibilità di risolvere economicamente e senza ulteriori costi di ricerca i problemi inerenti la trasmissione di dati per via ottica.

I trasmettitori dispongono, a seconda dell'esecuzione, di un trasduttore elettro-ottico, con diverse intensità luminose e di una elettronica che ha il compito di convertire l'informazione elettrica in entrata in correnti atte a modulare il diodo luminescente.

I ricevitori amplificano, possibilmente senza rumore e con elevata dinamica il segnale fornito dal diodo e lo adattano allo standard TTL.

I trasmettitori ed i ricevitori digitali le cui caratteristiche sono mostrate in fig. 51 (SIEMENS), permettono di realizzare linee di trasmissione con capacità fino a 5 Mbit/s in codice RZ.

velocità di trasmiss. RZ	trasmett. **V42253-**	U_B V	ricevitore **V42253-**	U_B V	cavo ottico **V42253-C1-**	lunghezza massima m
250 kbit/s	-G1 -B1-	+ 5	-H1-B2	± 5	-D..., fascio di fibre / 45 dB/100 m	37
250 kbit/s	-G1 -B1	+ 5	-H1-B5	± 12	-D..., fascio di fibre / 45 dB/100 m	37
250 kbit/s	-G1 -B2	+ 5	-H1-B2	± 5	-B..., profilo a gradino / 4 dB/100 m	390
250 kbit/s	-G1 -B2	+ 5	-H1-B5	± 12	-B..., profilo a gradino / 4 dB/100 m	390
250 kbit/s	-G1 -B3	+ 5	-H1-B2	± 5	-B.., profilo a gradino / 4 dB/100 m	640
250 kbit/s	-G1 -B3	+ 5	-H1-B5	± 12	-B..., profilo a gradino / 4 dB/100 m	640
250 kbit/s	-G1 -B3	+ 5	-H1-B2	± 5	-A..., profilo graduale / 1 dB/100 m	1300
250 kbit/s	-G1 - B3	+ 5	-H1-B5	± 5	-E.., profilo graduale / 0,5 dB/100 m	2600
250 kbit/s	-G -B3	+ 5	-H1-B5	± 12	-A..., profilo graduale / 1 dB/100 m	1300
250 kbit/s	-G1 -B3	+ 5	-H1-B5	± 12	-E..., profilo graduale / 0,5 dB/100m	2600

(segue tabella)

1 Mbit/s	-G -B1	+ 5	-H1-B4	± 12	-D..., fascio di fibre / 45 dB/100 m	32
1 Mbit/s	-G1 - B2	+ 5	-H1-B4	± 12	-B..., profilo a gradino / 4 dB/100 m	290
1 Mbit/s	-G1 - B3	+ 5	-H1-B4	± 12	-B..., profilo a gradino / 4 dB/100m	540
1 Mbit/s	-G1 -B3	+ 5	-H1-B4	± 12	-A..., profilo graduale / 1 dB/100 m	920
1 M/bits	-G1 -B3	+ 5	-H1-B4	± 12	-E..., profilo graduale / 0,5 db/100 m	1840
5 Mbit/s	-G1 - B1	+ 5	-H1 -B3	±12	-D..., fascio di fibre / 45 dB/100 m	18
5 Mbit/s	-G1 -B2	+ 5	-H1 -B3	± 12	-B..., profilo a gradino / 4 dB/100 m	185
5 Mbit/s	-G1 -B3	+ 5	-H1 -B3	± 12	-B..., profilo a gradino / 4 dB/100 m	435
5 Mbit/s	-G1 -B3	+ 5	-H1 -B3	± 12	-A..., profilo graduale / 1 dB/100 m	490
5 Mbit/s	-G1 -B3	+ 5	-H1 -B3	± 12	-E..., profilo graduale / 0,5 dB/100 m	980

FIG. 51. - Lunghezze di trasmissione raggiungibili con l'uso di trasmettitori e ricevitori digitali della serie VA42253 e cavi ottici di differente attenuazione.

I vari componenti possono essere combinati a piacere e vengono scelti in funzione delle esigenze della linea, di solito in base alla lunghezza ed alla larghezza della banda di trasmissione.

Nelle pagine seguenti vengono infine riportati le caratteristiche ed un circuito applicativo del trasmettitore ottico V 42253-G1-B1 e del ricevitore V 42253 -H1- B2, entrambi prodotti dalla SIEMENS, per una velocità di trasmissione fino a 1M bit/s in codice RZ e con un cavo di lunghezza massima di 37 metri.

Trasmettitore per cavi a più fibre V42253-G1-B1

Velocità di trasmissione: da 0 a 5 Mbit/s, codice RZ
Tensione d'esercizio: U_B = + 5V
ingresso dati: standard TTL

DATI TECNICI

tensione d'esercizio	U_B		5 +0,25		V
corrente assorbita	I_B		110		mA
ingresso dati	standard TTL		negativo		
corrente d'ingresso	I_H		100		μA
velocità di trasmissione	codice RZ codice NRZ		0 ÷ 5 0 ÷ 10		Mbit/s Mbit/s
lunghezza d'onda del segnale luminoso	λ		875		nm
coefficiente di temperatura dell'intensità luminosa	T_K		-0,6		%/K
temperatura ambiente	T_U		0 ÷ + 70		C°
intensità luminosa accoppiata con T_U = + 25° C nel cavo V 42253-C1-C...0 nel cavo V42253-C1-D...	Φ_S	min. 90	tipica 100	max 180	μW

andamento del segnale ottico a 5 Mbit/s (RZ)

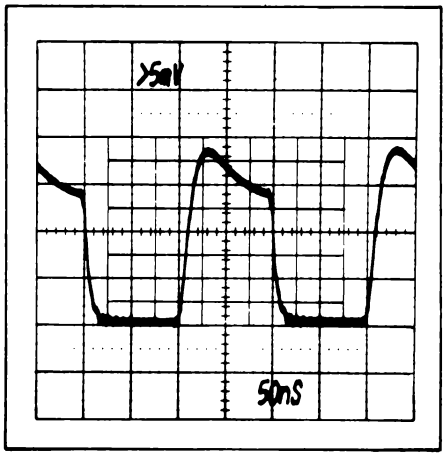


FIG. 52.-
Caratteristiche del trasmettitore V 42253 - G1 - B1 (SIEMENS).

ricevitore per tutti i tipi di cavi: V42253-H1-B2

Velocità di trasmissione: da 0 a 250 kbit/s, codice RZ
tensione d'esercizio: U_B = +5V
uscita dati: TTL e TTL

DATI TECNICI

tensione d'esercizio	U_B	±5±0,25	V
corrente assorbita	I_B (+ 5V) I_B (-5V)	20 11	mA mA
velocità di trasmissione	codice RZ codice NRZ	0 ÷ 250 0 ÷ 500	kbit/s kbit/s
corrente di uscita	I_H I_L	—1 10	mA mA
sensibilità con λ = 850 ± 30 nm	Φ_{Emin}	1,2	μW
limite di modulazione	Φ_{Emax}	110	μW
tasso di pulsazione a 250 kbit/s (codice RZ)	$t_1/t^1 + t_0$ t_1: durata della logica «1» t_0: durata della logica «0»	50 ± 8	%
temperatura ambiente	T_U	0 ÷ + 70	°C

FIG. 53. - Caratteristiche del ricevitore ottico V 42253 - H1 - B2 (SIEMENS).

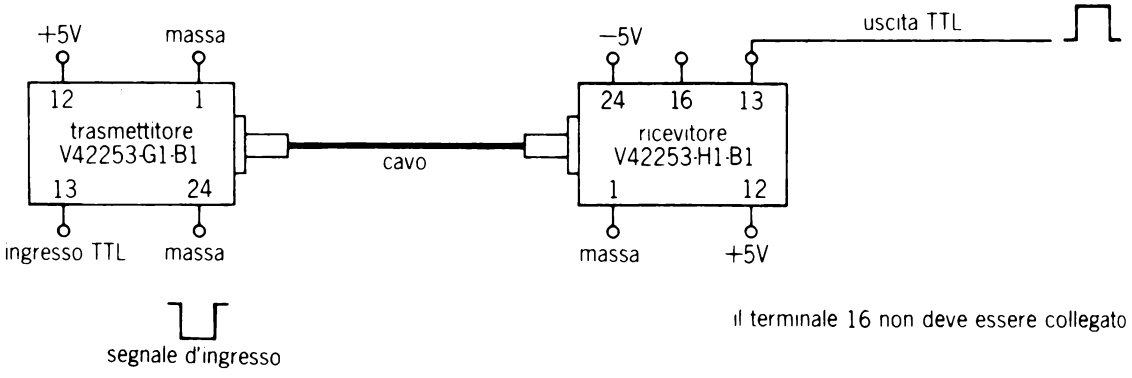


FIG. 54. - Trasmissione digitale delle informazioni tramite fibre ottiche (SIEMENS).

ESERCIZI PROPOSTI

1. Un bus è realizzato con collegamenti di tipo tri-state come è mostrato nella figura 55. È necessario porre dei resistori per la terminazione della linea?

2. Si determini l'impedenza caratteristica della linea dell'esercizio n° 1, sapendo che la CPU è la Z 80, che la linea è lunga 50 cm, che ha un'impedenza $Z_0 = 120\ \Omega$, e che i periferici DS 7833 sono in numero di 5.

3. Si determini il coefficiente di riflessione sul carico, di una linea alimentata a 10 MHz la cui impedenza caratteristica vale 120 Ω, ed il carico è costituito da una resistenza R_L 60 Ω con in parallelo un condensatore di capacità $C = 100$ pF.

4. Un generatore, di resistenza interna pari a 600 Ω, invia ad una linea di impedenza caratteristica $Z_0 = 300\ \Omega$ e lunghezza 1500 m chiusa su un carico di 10 KΩ, un impulso di ampiezza 5 V e durata 0,5 μs. Si disegni l'andamento della tensione agli estremi trasmittente e ricevente, usando il metodo dei coefficienti di riflessione.

5. Utilizzando il metodo grafico si calcolino i primi tre livelli di tensione all'estremità trasmittente e ricevente per una linea di impedenza caratteristica $Z_0 = 100\ \Omega$, chiusa su un carico di 1 KΩ ed alimentata con un gradino di tensione da un generatore di resistenza interna 50 Ω.

6. Come il numero 5 ma con un generatore di resistenza interna di 500 Ω.

7. Determinare le forme d'onda alle estremità trasmittente e ricevente in un sistema TTL, per la transizione da basso ad alto su una linea di impedenza $Z_0 = 50\ \Omega$.

8. Si disegni la reale forma d'onda del clock che si ottiene se l'impedenza della linea di massa di figura 29 vale 50 Ω mentre per la linea del segnale vale 100 Ω.

9. Si determini la massima lunghezza della connessione di massa per un circuito come quello di figura 29, considerando che il f-f può cambiare stato con un impulso di clock di 5 ns di durata e che si opera su piste di circuito stampato.

10. Determinare di quanto si deve ridurre la lunghezza dei collegamenti per la linea dell'esempio di pagina 236, se essa presenta una caduta ohmica di 30 mV per metro.

11. Supponendo uguale a 10dB/Km l'attenuazione specifica di un cavo a fibre ottiche, calcolare la lunghezza massima della linea di trasmissione tenendo presente che $\Phi_{5 = 30\,\mu W}$ e $\Phi_E = 0{,}5\ \mu$ W.

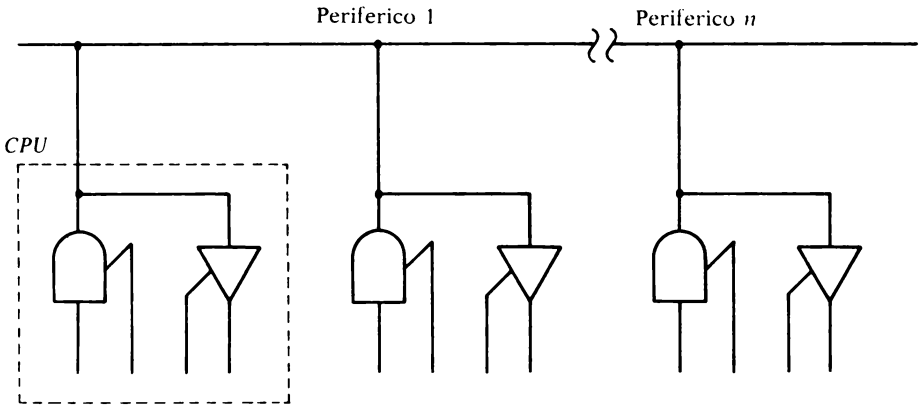


FIG. 55 - Bus tri-state.

CAPITOLO SESTO

APPLICAZIONI HARDWARE-SOFTWARE

In questo capitolo saranno esaminati dettagliamente alcuni dispositivi molto semplici ma concettualmente interessanti nelle loro funzioni svolte.

In particolare sarà affrontato e risolto il controllo di un braccio robotico a più gradi di libertà per mezzo di un microcomputer con unità centrale di elaborazione lo Z 80.

Naturalmente il lettore potrà effettuare pertinenti variazioni sia hardware che software per cercare di minimizzare il costo dei sistemi proposti dove ciò è possibile.

1. Acquisizione di dati da un Bus ad otto linee

Il problema trattato in questo paragrafo riguarda il progetto hardware di un dispositivo capace di realizzare l'acquisizione di dati provenienti da un bus ad otto linee (bus che può essere simulato per esempio con le uscite di due contatori binari collegati in cascata) effettuando la memorizzazione della successiva parola che lo percorre a partire da una parola di trigger (riconoscimento) impostata tramite switches.

Tale dispositivo può essere usato per controllare l'esatto funzionamento di un apparato molto veloce di cui si conosce la sequenzialità. Ad esempio se l'apparato sotto esame è un contatore binario e se la parola di trigger impostata è 00000010 in caso di corretto funzionamento il dato memorizzato dovrà essere uguale a 00000011.

In fig. 1 è mostrato lo schema a blocchi, il cui principio di funzionamento ricalca quello di un analizzatore di stati logici (cap. 9°) del dispositivo richiesto. Esso è costituito da una rete comparativa che riconosce la parola di trigger e da una memoria che acquisisce il dato successivo a tale parola.

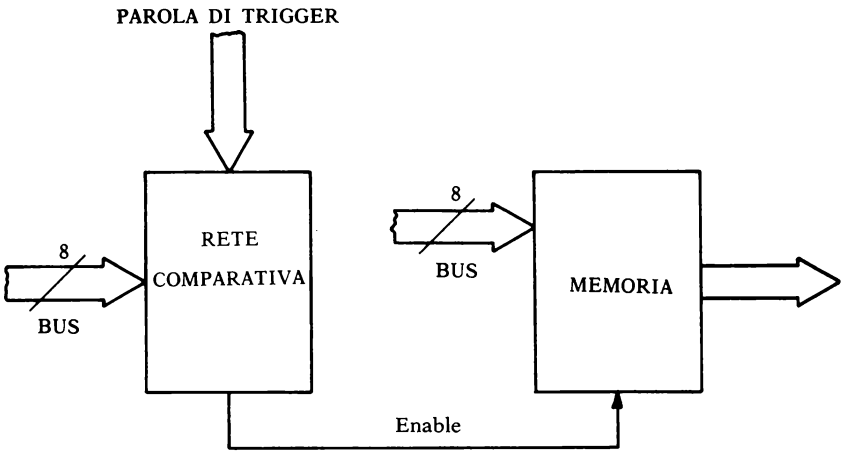


FIG. 1. - Schema a blocchi di un semplice analizzatore di stati logici.

La rete comparativa può essere realizzata o mediante porte logiche Xor ed Or o più semplicemente mediante il circuito integrato SN 74LS85 (Texas) le cui caratteristiche sono riportate in fig. 2.

Per la memoria possono essere utilizzati o flip-flop di tipo D p.e.t. oppure il porto di uscita SN 74LS374 la cui piedinatura e tabella della verità sono riportate nella fig. 62 del quarto capitolo.

Il funzionamento del circuito di fig. 3 è il seguente:

per mezzo dei circuiti comparatori vengono confrontati tutti i bit che percorrono il bus con i bit della parola di trigger precedentemente impostata.

Non appena la parola transitante sul bus risulta uguale a quella di trigger le uscite dei comparatori (ultima riga della prima tabella di fig. 2) commutano dal livello logico basso a quello alto; l'uscita della porta logica Nand, normalmente alta, collegata all'ingresso di clock del porto d'uscita commuta quindi al livello logico basso. Il dato presente agli ingressi D non viene memorizzato in quanto i flip-flop costituenti l'integrato sono del tipo p.e.t. e commutano quindi soltanto durante il fronte di salita del clock. La successiva parola a quella di trigger presente oltre che agli ingressi dei comparatori anche agli ingressi D della memoria viene poi immagazzinata in quanto durante questa fase si ha la transizione dal livello basso a quello alto dell'ingresso di clock.

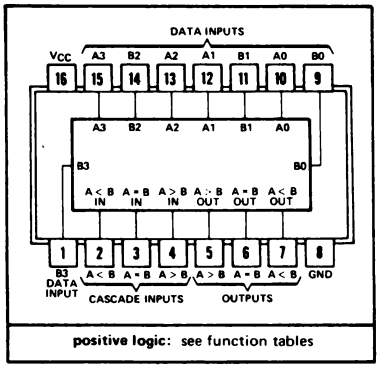
La successione dei segnali può essere schematizzata come in fig. 4.

TYPES SN5485, SN54L85, SN54LS85, SN54S85, SN7485, SN74L85, SN74LS85, SN74S85
4-BIT MAGNITUDE COMPARATORS

BULLETIN NO. DL-S 7611810, MARCH 1974—REVISED OCTOBER 1976

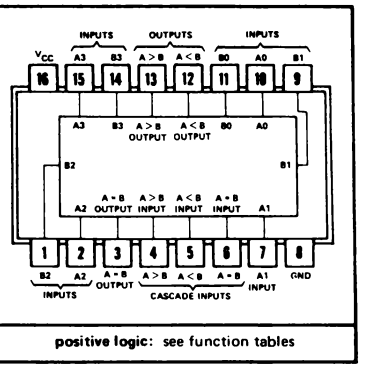
TYPE	TYPICAL POWER DISSIPATION	TYPICAL DELAY (4-BIT WORDS)
'85	275 mW	23 ns
'L85	20 mW	90 ns
'LS85	52 mW	24 ns
'S85	365 mW	11 ns

SN5485, SN54LS85, SN54S85 . . . J OR W PACKAGE
SN7485, SN74LS85, SN74S85 . . . J OR N PACKAGE
(TOP VIEW)



positive logic: see function tables

SN54L85 . . . J PACKAGE
SN74L85 . . . J OR N PACKAGE
(TOP VIEW)



positive logic: see function tables

description

These four-bit magnitude comparators perform comparison of straight binary and straight BCD (8-4-2-1) codes. Three fully decoded decisions about two 4-bit words (A, B) are made and are externally available at three outputs. These devices are fully expandable to any number of bits without external gates. Words of greater length may be compared by connecting comparators in cascade. The A > B, A < B, and A = B outputs of a stage handling less-significant bits are connected to the corresponding A > B, A < B, and A = B inputs of the next stage handling more-significant bits. The stage handling the least-significant bits must have a high-level voltage applied to the A = B input and in addition for the 'L85, low-level voltages applied to the A > B and A < B inputs. The cascading paths of the '85, 'LS85, and 'S85 are implemented with only a two-gate-level delay to reduce overall comparison times for long words. An alternate method of cascading which further reduces the comparison time is shown in the typical application data.

FUNCTION TABLES

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B2	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H	L	L	H

'85, 'LS85, 'S85

A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

'L85

A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	H	L	H	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	H	H	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	H	H	H	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	H	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	L	L	L

H = high level, L = low level, X = irrelevant

FIG. 2.-
Caratteristiche del 74LS85.

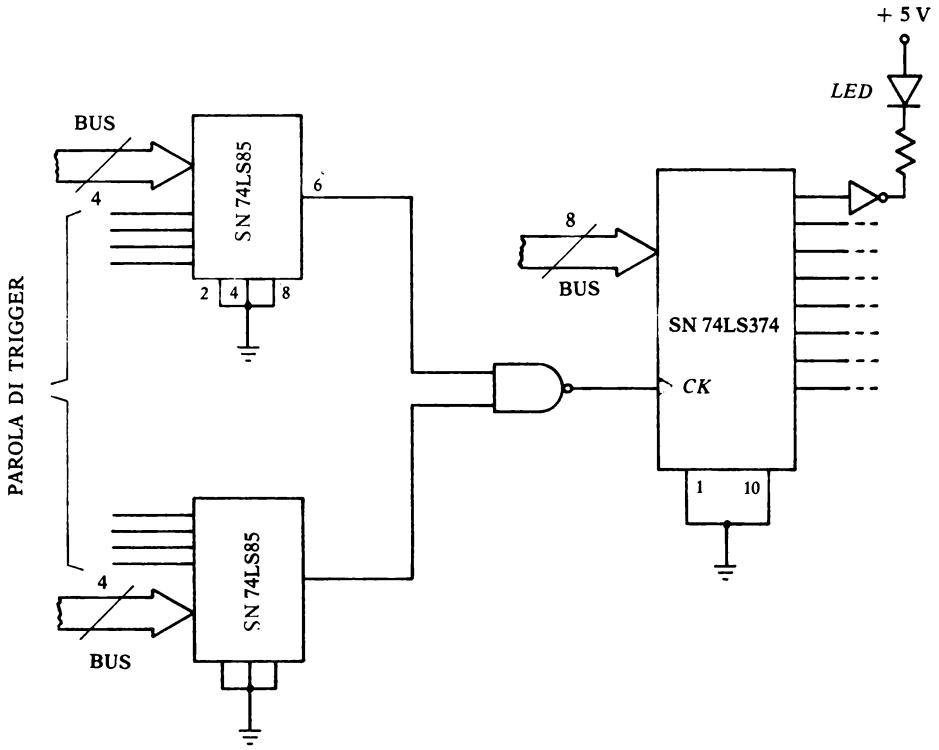


FIG. 3. - Schema completo di un semplice analizzatore di stati logici.

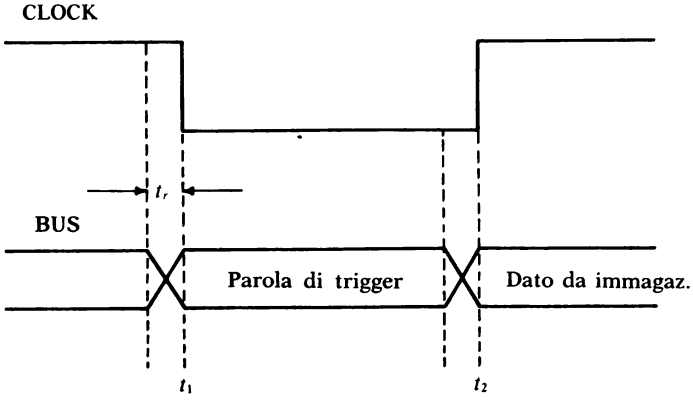


FIG. 4. -

Al tempo t_1 la rete comparativa riconosce la parola già presente da un tempo t_r (tempo di ritardo della rete comparativa) nel bus. La parola successiva da memorizzare viene riconosciuta al tempo t_2 e non essendo la parola di trigger fa commutare l'uscita del Nand dal livello logico 0 al livello logico 1. Questo fronte di salita abilita i flip-flop ad immagazzinare il dato.

La verifica del corretto funzionamento del circuito può essere effettuata visualizzando il contenuto degli otto flip-flop costituenti la memoria per mezzo di otto diodi LED guidati da inverter.

Il valore della resistenza R, al solito, deve poter limitare a 16 mA la corrente I_{0L} assorbita dall'inverter quando la sua uscita viene portata al livello logico basso.

2. Contatore guidato a microprocessore

Si è già visto nel capitolo quinto del volume di Elettronica Digitale come per passare da un contatore a modulo n ad uno a modulo m (con $n \neq m$) occorra modificare l'hardware del sistema. In questo paragrafo si vedrà invece come restando fisso l'hardware del sistema si possa ottenere un contatore a modulo variabile oppure un cronometro digitale agendo soltanto sul software.

Per semplicità sarà esaminato un dispositivo a due cifre tenendo presente che le stesse considerazioni restano valide anche per un analogo dispositivo con un numero qualsiasi di cifre.

Il circuito di interfaccia usato per la realizzazione del contatore è quello di fig. 6. La tecnica di gestione di I/O è quella di I/O isolato.

Le funzioni svolte dai singoli circuiti integrati sono le seguenti:

SN 7432

Abilita il demultiplexer. Ciò avviene quando $\overline{IORQ}$ e $\overline{WR}$ diventano bassi per effetto di una istruzione di uscita (fig. 5).

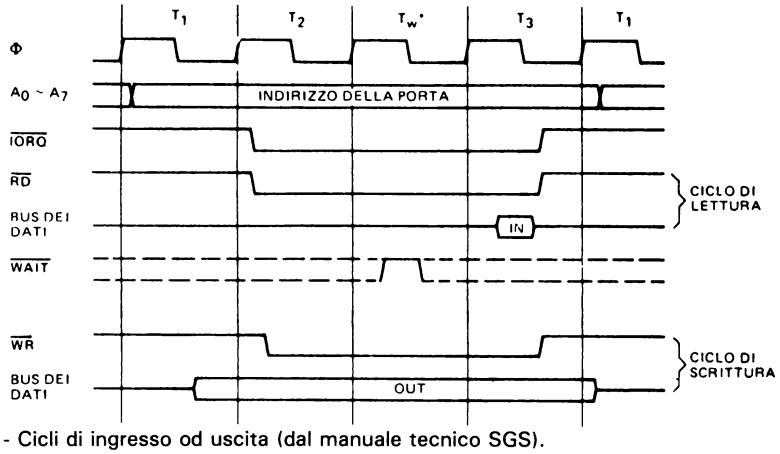


FIG. 5. - Cicli di ingresso od uscita (dal manuale tecnico SGS).

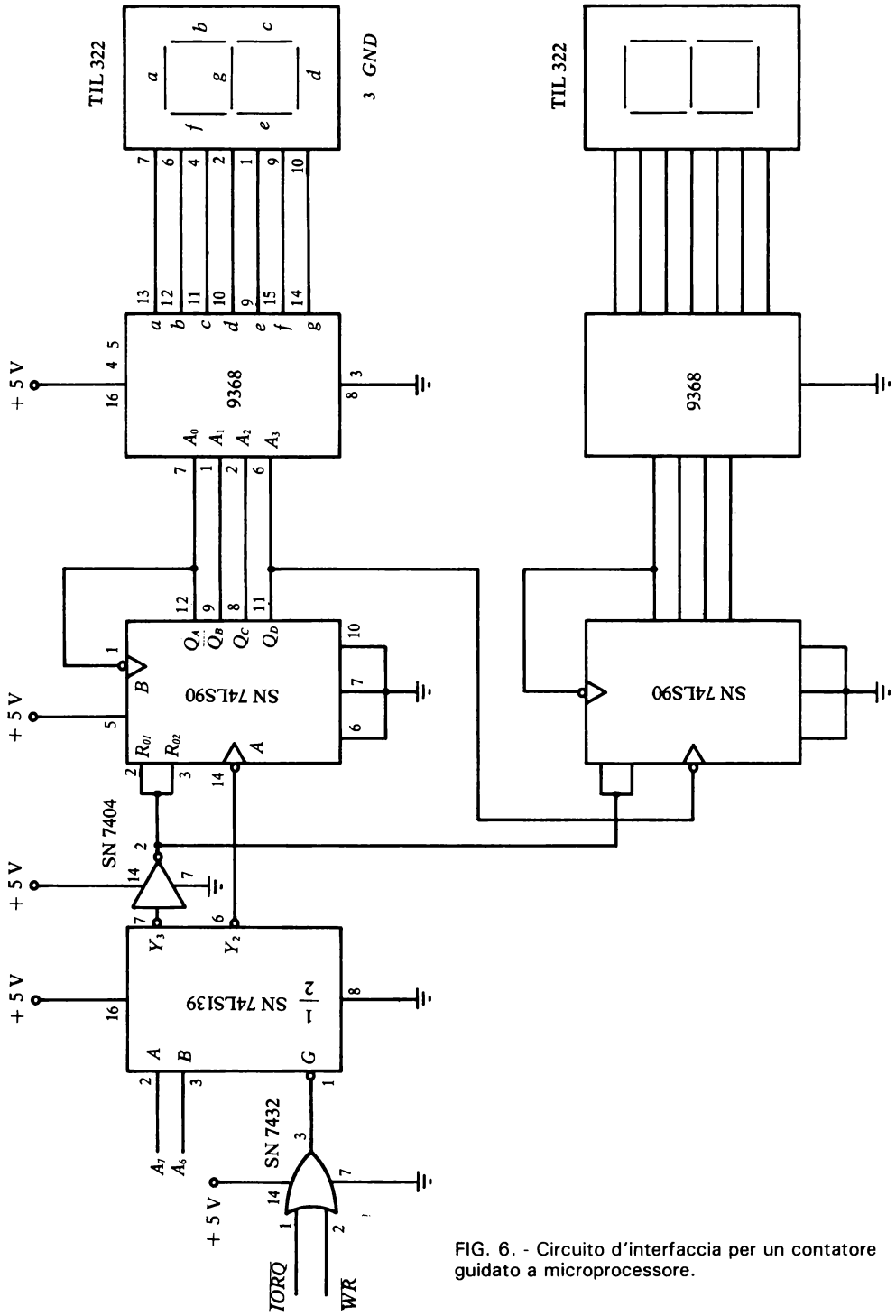


FIG. 6. - Circuito d'interfaccia per un contatore guidato a microprocessore.

74LS139

Viene usato per selezionare, tramite le linee d'indirizzo A_6 ed A_7, il dispositivo d'uscita desiderato. La tabella funzionale del demultiplexer già mostrata in fig. 9 del quarto capitolo viene riportata in fig. 7 per comodità.

'LS139, 'S139
(EACH DECODER/DEMULTIPLEXER)
FUNCTION TABLE

INPUTS			OUTPUTS			
ENABLE	SELECT					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = high level, L = low level, X = irrelevant

FIG. 7. - Tabella funzionale dell'SN 74LS139

SN 7404

Permette l'azzeramento del contatore

SN 74LS90

Effettuano il conteggio in BCD degli impulsi entrati nel dispositivo.

9368

Permettono la decodifica da BCD a sette segmenti.

TIL 322

Visualizzano il conteggio.

Il funzionamento del circuito di fig. 6 è il seguente:
la prima istruzione di OUT (*) seleziona l'uscita y_3 in quanto in queste condizioni sulla parte bassa del bus degli indirizzi è posto il dato 11000000 = $C0_H$ (contenuto del registro C) mentre $\overline{IORQ}$ e $\overline{WR}$ vanno bassi simultaneamente abilitando così il demultiplexer. Per non complicare eccessivamente il circuito di decodifica si sono utilizzate soltanto due linee d'indirizzo, ciò naturalmente comporta un'abilitazione dell'SN 74LS139 indipendentemente dal valore assunto dalle rimanenti linee d'indirizzo.

(*) La descrizione di questa istruzione è riportata nel paragrafo 3.

Il dato C0 in uscita seleziona quindi y_3 in quanto gli ingressi del demultiplexer offrono la seguente combinazione:

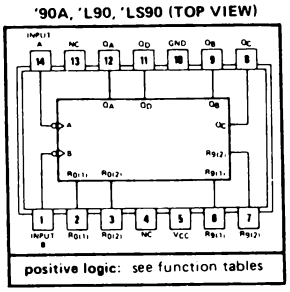
G	B	A
0	1	1

L'uscita del demultiplexer, normalmente alta, viene portata al livello logico basso e gli ingressi R_{01} ed R_{02} dei contatori tramite l'inverter sono portati al livello logico alto permettendo così l'azzerramento del dispositivo (fig. 8).

Successivamente mediante un'altra istruzione di OUT (ad esempio caricando nel registro C il codice dispositivo 0100000 = 40_H) si seleziona l'uscita y_2 che, collegata all'ingresso di clock del contatore, fornisce un impulso di conteggio.

Inserendo nel programma un loop di ritardo è possibile far contare il contatore BCD fino alla configurazione desiderata ed alla velocità voluta, oppure mantenerlo sempre in fase di conteggio.

'90A, 'L90, 'LS90 (TOP VIEW)



positive logic: see function tables

'90A, 'L90, LS90
BCD COUNT SEQUENCE
(See Note A)

COUNT	OUTPUT			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H

'90A, 'L90, 'LS90
BI-QUINARY (5-2)
(See Note B)

COUNT	OUTPUT			
	Q_A	Q_D	Q_C	Q_B
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	H	L	L	L
6	H	L	L	H
7	H	L	H	L
8	H	L	H	H
9	H	H	L	L

'90A, 'L90, 'LS90
RESET/COUNT FUNCTION TABLE

RESET INPUTS				OUTPUT			
$R_{0(1)}$	$R_{0(2)}$	$R_{9(1)}$	$R_{9(2)}$	Q_D	Q_C	Q_B	Q_A
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
X	L	X	L	COUNT			
L	X	L	X	COUNT			
L	X	X	L	COUNT			
X	L	L	X	COUNT			

NOTES: A. Output Q_A is connected to input B for BCD count.
B. Output Q_D is connected to input A for bi-quinary count.
C. Output Q_A is connected to input B.
D. H = high level, L = low level, X = irrelevant

FIG. 8. - Piedinatura e tabella della verità dell'SN 74LS90 (Texas).

9368 (FAIRCHILD)

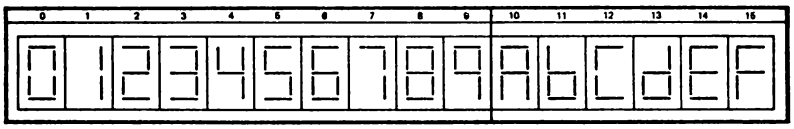
DESCRIPTION — The 9368 is a TTL/MSI Seven Segment Decoder/Driver incorporating input latches, and output circuits to drive common cathode type LED displays directly.

- **HIGH SPEED INPUT LATCHES FOR DATA STORAGE**
- **DRIVES COMMON CATHODE LED DISPLAYS DIRECTLY**
- **ACTIVE LOW LATCH ENABLE FOR EASY INTERFACE WITH MSI CIRCUITS**
- **HEXADECIMAL DECODE FORMAT**
- **LATCH SPEED COMPARABLE TO STANDARD MSI LATCHES**
- **DATA INPUT FAN IN ZERO WHEN LATCH NOT ENABLED***
- **AUTOMATIC RIPPLE BLANKING FOR SUPPRESSION OF LEADING EDGE ZEROS AND/OR TRAILING EDGE ZERO's**
- **PINOUT COMPARABLE WITH OTHER STANDARD MSI DECODERS SUCH AS 9307, 9317, 9357A (7446), 9357B (7447), 9358 (7448), 9359 (7449)**

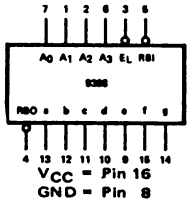
PIN NAMES		LOADING HIGH	LOADING LOW
A_0, A_1, A_2, A_3	Address Inputs	2.0 U.L.	1.0 U.L.*
$\overline{E_L}$	Latch Enable	1.0 U.L.	1.0 U.L.
$\overline{RBI}$	Ripple Blanking (Active LOW) Input	1.0 U.L.	1.0 U.L.
$\overline{RBO}$	Ripple Blanking (Active LOW) Output	2.0 U.L.	2.0 U.L.
a, b, c, d, e, f, g	(Active HIGH) Outputs	20 mA	"OFF"

*LOW level loading is 1 U.L. only when latch is enabled. When latch is disabled loading is 10 μA.

NUMERICAL DESIGNATIONS



LOGIC SYMBOL



CONNECTION DIAGRAM
16-LEAD DIP
(TOP VIEW)
PACKAGE OUTLINE 6B

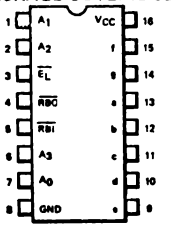


FIG. 9. - Piedinatura del 9368 (Fairchild)

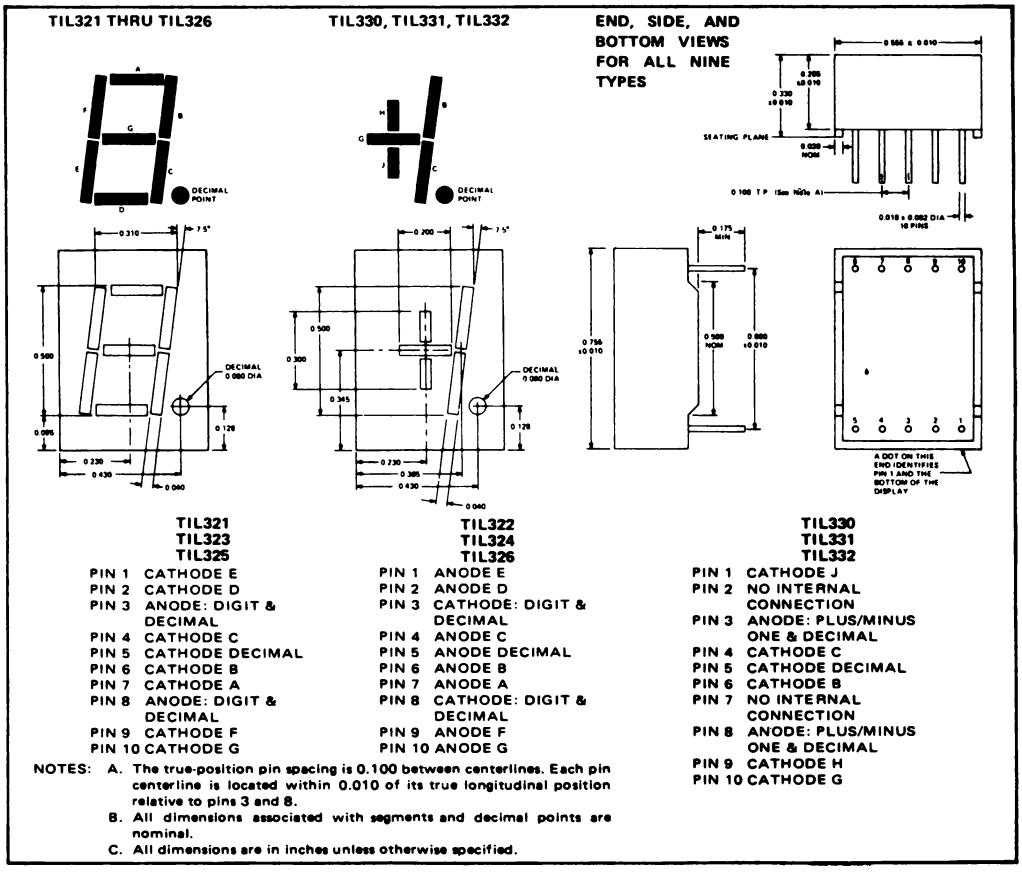
SOLID-STATE VISIBLE DISPLAYS WITH RED, GREEN, OR AMBER TRANSPARENT PLASTIC ENCAPSULATION

- 0.500-Inch-High Characters
- Continuous Uniform Segments
- Wide Viewing Angle
- High Contrast
- Categorized for Uniformity of Luminous Intensity among Units within Each Category
- Low Power Requirements

	SEVEN SEGMENTS WITH RIGHT DECIMAL, COMMON ANODE	SEVEN SEGMENTS WITH RIGHT DECIMAL, COMMON CATHODE	PLUS/MINUS ONE WITH RIGHT DECIMAL
RED	TIL321	TIL322	TIL330
GREEN	TIL323	TIL324	TIL331
AMBER	TIL325	TIL326	TIL332*

mechanical data

The light-emitting diode chips are mounted on a printed-circuit board, which, together with a one-piece reflector assembly, is encased within a transparent plastic case and sealed with epoxy. To optimize device performance, materials are used that are limited to certain solvents for cleaning operations. It is recommended that only freon TF, isopropanol, or water be used.



TIL321 TIL323 TIL325

PIN 1 CATHODE E
PIN 2 CATHODE D
PIN 3 ANODE: DIGIT & DECIMAL
PIN 4 CATHODE C
PIN 5 CATHODE DECIMAL
PIN 6 CATHODE B
PIN 7 CATHODE A
PIN 8 ANODE: DIGIT & DECIMAL
PIN 9 CATHODE F
PIN 10 CATHODE G

TIL322 TIL324 TIL326

PIN 1 ANODE E
PIN 2 ANODE D
PIN 3 CATHODE: DIGIT & DECIMAL
PIN 4 ANODE C
PIN 5 ANODE DECIMAL
PIN 6 ANODE B
PIN 7 ANODE A
PIN 8 CATHODE: DIGIT & DECIMAL
PIN 9 ANODE F
PIN 10 ANODE G

TIL330 TIL331 TIL332

PIN 1 CATHODE J
PIN 2 NO INTERNAL CONNECTION
PIN 3 ANODE: PLUS/MINUS ONE & DECIMAL
PIN 4 CATHODE C
PIN 5 CATHODE DECIMAL
PIN 6 CATHODE B
PIN 7 NO INTERNAL CONNECTION
PIN 8 ANODE: PLUS/MINUS ONE & DECIMAL
PIN 9 CATHODE H
PIN 10 CATHODE G

NOTES: A. The true-position pin spacing is 0.100 between centerlines. Each pin centerline is located within 0.010 of its true longitudinal position relative to pins 3 and 8.
B. All dimensions associated with segments and decimal points are nominal.
C. All dimensions are in inches unless otherwise specified.

FIG. 10. - Piedinatura e caratteristiche di alcuni tipi di display (Texas).

Il diagramma a blocchi ed il software relativo per il funzionamento del circuito contatore sono mostrati rispettivamente in fig. 11 e nel programma che segue.

Flow-chart:

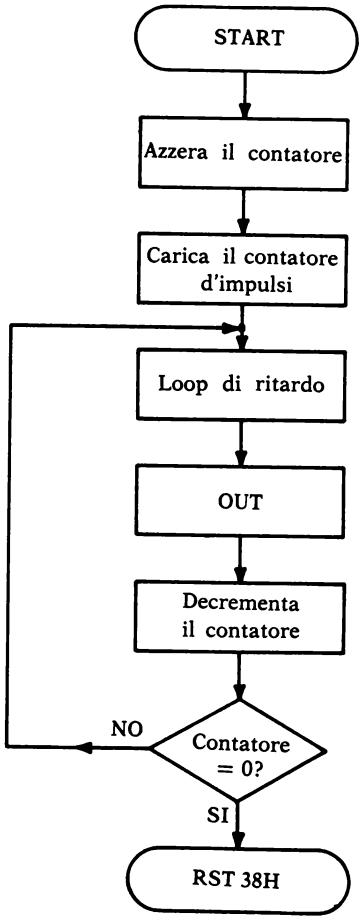


FIG. 11. -

Programma:

	LD C, C0	Carica il codice dispositivo di azzeramento (11000000)
	OUT (C), C	Azzera il contatore ($\overline{IORQ} = \overline{WR} = 0$, $A_7 = A_6 = 1$). Impulso di uscita su y_3
	LD E, n	Carica il numero degli impulsi da contare
P_4	LD C, n	
P_3	LD L, n	
P_2	LD H, n	
P_1	DEC H	Loop di DELAY
	JP NZ P_1	
	DEC L	
	JP NZ P_2	
	DEC C	
	JP NZ P_3	
	LD C, 40	Carica il codice dispositivo di conteggio (01000000).
	OUT (C), C	Genera un impulso di uscita su y_2 ($\overline{IORQ} = \overline{WR} = 0$, $A_7 = 0, A_6 = 1$).
	DEC E	
	JP NZ P_4	Salta a P_4 se il contatore è diverso da zero
	RST 38 H	

La frequenza di conteggio può essere fissata di caso in caso variando il contenuto dei registri C,L ed H nel loop di delay. Il numero degli impulsi da contare viene caricato nel registro E. È consigliato oltre che per motivi di potenza anche per motivi di costo l'uso di integrati LS in quanto attualmente tali integrati per la loro maggiore richiesta rispetto a quelli con tecnologia TTL normale risultano economicamente più convenienti.

Riferendosi al flow-chart di fig. 11 possono essere fatte due diverse considerazioni.

1) Volendo realizzare, ad esempio, un cronometro al centesimo di secondo basterà calcolare il loop di delay in modo di avere un impulso di uscita ogni 0,01 secondi e caricare nel registro contatore l'esadecimale 64 (100 in decimale).

2) Se si desidera realizzare invece un contasecondi occorrerà caricare nel registro E l'esadecimale 3C (60 in decimale) ricaricarlo ogni volta che si azzera spostando il salto condizionato verso l'alto di un blocco funzionale, e fissare la durata del loop di delay ad un secondo.

Mettendo infine in Nand le uscite Q_B e Q_C del decodificatore di decine e collegando l'uscita del Nand all'ingresso A di un circuito come quello utilizzato per il conteggio dei secondi è possibile contare i minuti, e così via per le ore.

3. Acquisizione dati mediante microcomputer

Un classico esempio di controllo ad anello chiuso mediante microcomputer è mostrato in fig. 12.

Il dispositivo comanda una lampada ad incandescenza ed un ventilatore usati rispettivamente per riscaldare o raffreddare un ambiente chiuso in cui la temperatura è controllata da un trasduttore di temperatura o sensore.

L'intensità luminosa della lampada così come l'accensione o meno del ventilatore sono controllate mediante il software da un microcomputer in modo da mantenere costante il valore della temperatura nel luogo considerato.

Il multiplexer analogico permette di interrogare all'occorrenza lo stato di altri sette sensori collocati in altrettanti punti di misura. Quando il microprocessore vuole conoscere la temperatura in un certo punto dell'ambiente abilita il multiplexer, seleziona il sensore corrispondente e immagazzina in dei latches l'indirizzo che lo identifica per tutto il tempo necessario alla conversione della grandezza.

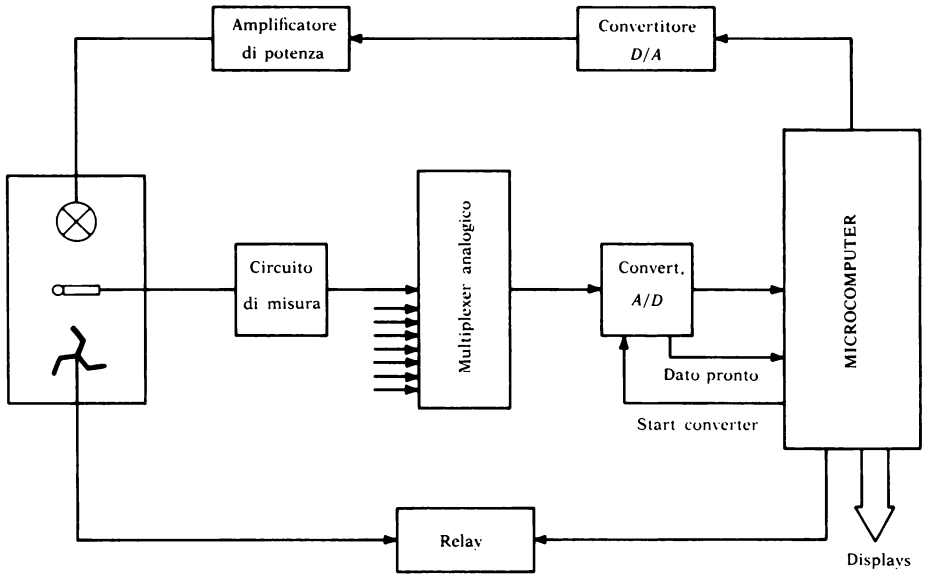


FIG. 12. - Esempio di controllo a catena chiusa.

La conversione ha inizio non appena il microprocessore invia un impulso di *initiate conversion* (start converter) al convertitore A/D ed ha termine quando quest'ultimo invia alla CPU un segnale di *dato pronto*. A questo punto il microprocessore legge ed interpreta il dato ed eventualmente lo visualizza su dei display a sette segmenti effettuando la decodifica da BCD a decimale. Se il valore letto si discosta da quello prefissato il microprocessore agisce sulla lampada (tramite un convertitore digitale-analogico e un amplificatore di potenza) o sull'attivazione del ventilatore (tramite un relay) oppure su entrambi in modo da riportare la temperatura al valore desiderato.

Il circuito di misura all'ingresso analogico del multiplexer, che può essere un amplificatore o un'attenuatore, ha la funzione di rendere compatibili i livelli di tensione provenienti dal trasduttore di temperatura con quelli del multiplexer.

In fig. 14 è mostrato lo schema a blocchi di una parte del dispositivo e precisamente la parte riguardante l'acquisizione dati limitata alla gestione di un solo sensore distante non più di qualche decina di metri dalla parte ricevente.

Per distanze maggiori occorre convertire la tensione rilevata dal sensore in una corrente ed inviare quest'ultima al ricevitore mediante un trasmettitore *Two Wire* come mostrato in fig. 13 (dal manuale tecnico National).

La conversione inizia su comando della CPU inviando un impulso all'ingresso di start conversion del convertitore A/D 8700 della Teledyne Semiconductor le cui caratteristiche sono mostrate nelle fig. 15, 16, 17, 18 e 19.

Terminata la conversione, l'uscita *dato valido* dell'8700 genera una richiesta di interruzione, la routine di servizio dell'interrupt salva il contenuto dei registri della CPU nello stack e legge il dato presente sulle linee di uscita dell'SN 74LS244.

La selezione del porto d'ingresso (tri state), del porto di uscita (flip-flop A) e del bus bidirezionale viene effettuata mediante la tecnica di I/0 isolato decodificando per semplicità soltanto la linea A_7 del bus degli indirizzi cosicchè qualunque parola il cui bit più significativo è uguale ad uno effettuerà la selezione voluta (per es. FF).

Non appena vengono eseguite le istruzioni LDA, 7F ed OUT (FF), A il bus bidirezionale SN 74LS245, le cui caratteristiche sono quelle di fig. 20, viene abilitato ($\overline{IORQ} = \overline{WR} = DIR = 0$, $A_7 = 1$) verso sinistra ed il contenuto dell'accumulatore 7F = $0111\ 1111_2$ viene messo sul bus dei dati. In questa situazione l'ingresso di clock del flip-flop B commuta dal livello logico basso a quello alto e porta ad 1 l'uscita Q ($D_7 = D = 0$) fornendo un segnale di inizio conversione per l'8700.

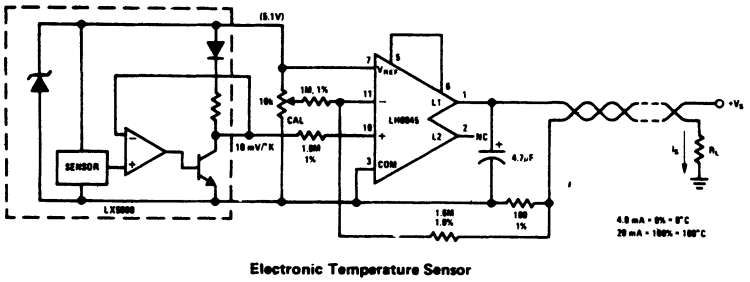


FIG. 13. - Collegamento di un sensore di temperatura con la parte ricevente a media-lunga distanza tramite l'LH0045 (dal manuale tecnico National).

Poichè una istruzione di uscita nel caso dello Z 80 per essere eseguita ha bisogno di 12 cicli di clock, supponendo la frequenza di lavoro della CPU uguale a 2,5 MH_Z, il tempo di esecuzione vale 4,8 μs, di conseguenza l'impulso di initiate conversion dovrà almeno avere la stessa durata.

Mediante poi l'esecuzione delle istruzioni LD A, FF e OUT (FF), A ($D_7 = D = 1$) l'ingresso di initiate conversion viene riportato basso. A questo punto il microprocessore è libero di eseguire altre operazioni di processo.

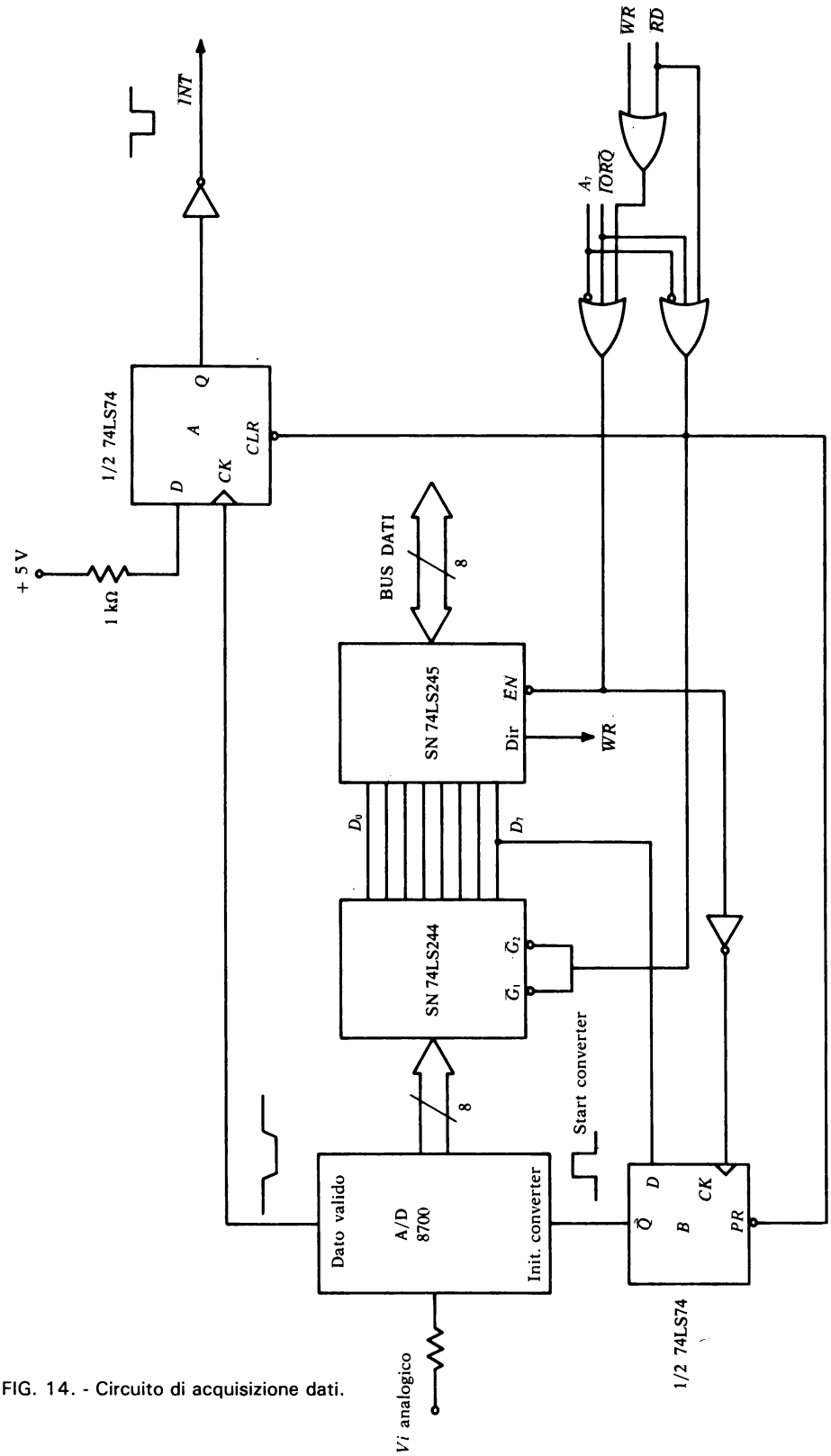


FIG. 14. - Circuito di acquisizione dati.

Monolithic CMOS A/D Converters
8700 Series

8700, 8701, 8702
8, 10, 12 Bit
Binary

Features

High Accuracy – Up to 12 Bit Resolution with $< \pm\frac{1}{2}$ LSB Error

Monotonic Performance – No Missing Codes

Monolithic CMOS Construction Gives Low Power Dissipation – 20mW Typical

Contains All Required Active Elements – Needs Only Passive Support Components, Reference Voltage And Dual Power Supply

High Stability Over Full Temperature Range
- Gain Temperature Coefficient Typically <25ppm/°C
- Zero Drift Typically <30μV/°C
- Differential Non-Linearity Drift Typically <2.5ppm/°C

Latched Parallel Binary Outputs

LPTTL And CMOS Compatible Outputs And Control Inputs

Strobed Or Free Running Conversion

Infinite Input Range – Any Positive Voltage Can Be Applied Via A Scaling Resistor

Absolute Maximum Ratings

Storage Temperature		−65°C to +150°C
Operating Temperature	(N Package)	−40°C to +85°C
	(J Package)	0°C to 70°C
$V_{DD} - V_{SS}$		18V
I_{IN}		±10mA
I_{REF}		±10mA
Digital Input Voltage		−0.3 to V_{DD} +0.3V
Operating V_{DD} and V_{SS} Range		**3.5V to 7V**
Package Dissipation		500mW
Lead Temperature (Soldering, 10 seconds)		300°C

HANDLING PRECAUTIONS

The 8700 series are CMOS devices and must be handled correctly to prevent damage. Package and store only in conductive foam, anti-static tubes or other conductive material. Use proper anti-static handling procedures. Do not connect in circuits under "power on" conditions, as high transients may cause permanent damage.

General Description

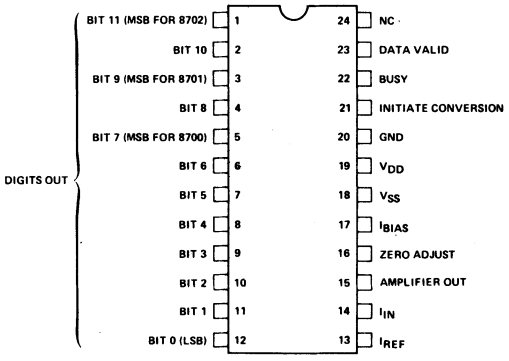
The Teledyne Semiconductor 8700/8701/8702 are 8/10/12 bit monolithic CMOS analog-to-digital converters. Fully self-contained in a single 24-pin dual in-line package, each converter requires only passive support components, voltage or current reference and power supplies.

Conversion is performed by an incremental charge balancing technique which has inherently high accuracy, linearity and noise immunity. An amplifier integrates the sum of the unknown analog current and pulses of a reference current, and the number of pulses (charge increments) needed to maintain the amplifier summing junction near zero is counted. At the end of conversion the total count is latched into the digital outputs as an 8/10/12 bit binary word.

Connection Diagram

Order Part Numbers:

N Package	**J Package**
24-Pin Ceramic DIP	**24-Pin Plastic DIP**
(−40°C to +85°C)	**(0°C to 70°C)**
8700CN – 8 bit	8700CJ – 8 bit
8701CN – 10 bit	
8702CN – 12 bit	



NOTE: Pin 1 indicated by adjacent dot or indent (N package), or end notch (J package). Do not make connections to pin 1, 2, 3 or 4 on 8700; pin 1 or 2 on 8701. These pins are internally connected.

FIG. 15. - Piedinatura e caratteristiche della serie 8700 (Teledyne).

Electrical Characteristics Unless otherwise specified, V_{DD} = +5V, V_{SS} = −5V, V_{GND} = 0, V_{REF} = −6.4V, R_{BIAS} = 100KΩ, test circuit shown. T_A = 25°C unless Full Temp. Range is specified (−40°C to +85°C for N package, 0°C to 70°C for J package).

Parameter	Definition	Min	Typ	Max	Units	Conditions
Accuracy						
Resolution Accuracy	Binary word length of digital output					
	8700	8			Bits	
	8701	10			Bits	
	8702	12			Bits	
Relative Accuracy	Output deviation from straight line between normalized zero and full scale input			±1/2	LSB	
Differential Non-Linearity	Deviation from 1 LSB between transition points		±1/4	±1/2	LSB	
Differential Non-Linearity Temperature Drift	Variation in Differential Non-Linearity due to temperature change		±2.5	±5	ppm/°C	Full Temp. Range
Gain Variance	Variation from exact A (compensate by trimming R_{IN} or R_{REF})		±2	+5 −3	% of Nominal	
Gain Temperature Drift	Variation in A due to temperature change		±25	±75	ppm/°C	Full Temp. Range
Zero Offset	Correction at zero adjust to give zero output when input is zero		±10	±50	mV	I_{IN} = 0
Zero Temperature Drift	Variation in zero offset due to temperature change		±30	±50	µV/°C	Full Temp. Range
Analog Inputs						
I_{IN} Full Scale	Full scale analog input current to achieve specified accuracy		10		µA	
I_{REF} (Note 1)	Reference current input to achieve specified accuracy		−20		µA	
Digital Inputs						
V_{IN} (1)	Logical "1" input threshold for Initiate Conversion Input	3.5			V	Full Temp. Range
V_{IN} (0)	Logical "0" input threshold for Initiate Conversion input			1.5	V	Full Temp. Range
Digital Outputs						
V_{OUT} (1)	Logical "1" output voltage for Digits Out, Busy, and Data Valid Outputs					Full Temp. Range
		4.5			V	I_{OUT} = −10µA
		2.4			V	I_{OUT} = −360µA
V_{OUT} (0)	Logical "0" output voltage for Digits Out, Busy, and Data Valid Outputs			0.4	V	Full Temp. Range V_{DD} = 4.75V, I_{OUT} = 360µA
Dynamic						
Conversion Time	Time required to perform one complete A/D conversion 8700		1.25	1.8	ms	Full Temp. Range
	8701		5	6	ms	
	8702		20	24	ms	
Conversion Rate in Free-Run Mode	8700	555	800		Conv'ns per Second	$V_{INIT\ CONV}$ = +5V
	8701	167	200			
	8702	42	50			
Minimum Pulse Width for Initiate Conversion		500			ns	Full Temp. Range
Supply Current						
I_{DD} Quiescent	Current required from positive supply during operation					
(N Package)			1.4	2.5	mA	
(J Package)			1.4	5.0	mA	Full Temp. Range
I_{SS} Quiescent	Current required from negative supply during operation					$V_{INIT\ CONV}$ = 0V
(N Package)			−1.6	−2.5	mA	
(J Package)			−1.6	−5.0	mA	
Supply Sensitivity	Change in full scale gain vs supply voltage change		±0.5	±1.0	%/V	V_{DD} ± 1V, V_{SS} ± 1V
	Change in full scale gain vs supply voltage change for tracking supplies		±0.05	±0.1	%/V	$\|V_{DD}\| = \|V_{SS}\|$ = 5V ± 1V

NOTE: I_{IN} and I_{REF} pins connect to the summing junction of an operational amplifier. Voltage sources cannot be attached directly but must be buffered by external resistors. See Test Circuit.

FIG. 16. - Caratteristiche tecniche.

Test Circuit

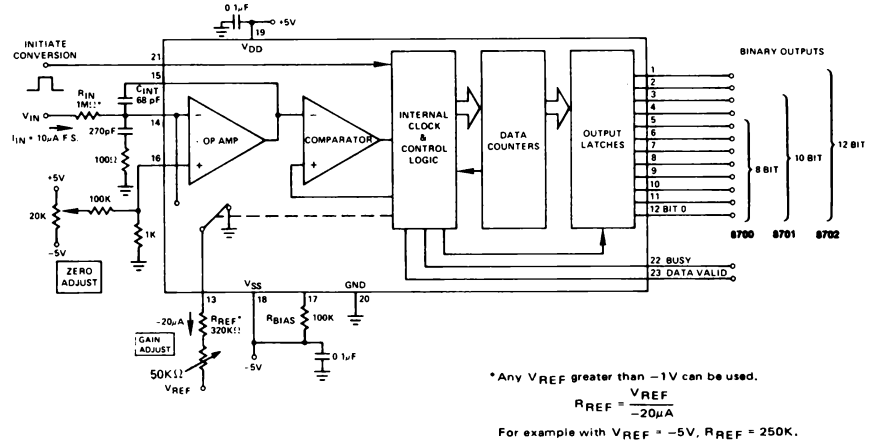


FIG. 17. - Schema a blocchi del convertitore A/D

Timing Diagrams (Rise, fall times = 200ns typ., C_L = 50pF)

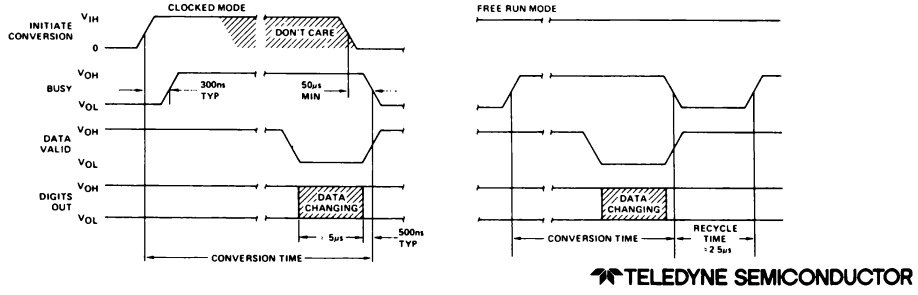


FIG. 18. - Diagrammi temporali.

Pin Functions

Initiate Conversion Input – Accepts CMOS and most 5V logic inputs. Applying a logic "1" to the Initiate Conversion pin initiates the A/D conversion cycle. Once conversion has been initiated, the cycle cannot be interrupted, and the Initiate Conversion pin is disabled until conversion is complete. Two modes of operation are permitted, clocked or free-running. For clocked operation the Initiate Conversion input is held at logic "0" for standby and taken to logic "1" when a conversion is desired. For free-running operation the Initiate Conversion pin is connected to V_{DD} or similar permanent logic "1" voltage.

Busy Output – A digital status output which is compatible with CMOS logic and low power TTL (can sink and source 500μA). A logic "1" output on the Busy pin indicates a conversion cycle is in process. A logic "1" to logic "0" transition indicates that conversion is complete and the result has been latched at the Digits Out pins. A logic "0" to logic "1" transition indicates a new conversion cycle has been initiated. If the device is operating in the free-running mode, the Busy output will remain low for approximately 2.5μs, marking the completion and initiation of consecutive conversion cycles.

Data Valid Output – A digital status which is compatible with CMOS logic and low power TTL (can sink and source 500μA). A logic "1" output at the Data Valid pin indicates that the Digits Out pins are latched with the result of the last conversion cycle. The Data Valid output goes to logic "0" approximately 5μs before the completion of a conversion cycle. During this 5μs interval new data is being transferred to the Digits Out pins, and the Digits Out are not valid.

Digits Out (Bit 0, Bit 1, etc.) – The binary digit outputs which are the result of the A/D conversion. These outputs are CMOS logic and low power TTL compatible.

Applications Information

Input/Output Relationships – The analog input voltage (V_{IN}) is related to the output by the transfer equation:

$$\text{DIGITAL COUNTS} = \frac{V_{IN} \cdot A \cdot R_{REF}}{R_{IN} \cdot V_{REF}}$$

A = 528 for 8700
A = 2064 for 8701
A = 8208 for 8702

where DIGITAL COUNTS is the value of the binary output word presented at Digits Out pins in response to V_{IN}.

The digital output code format is as follows:

ANALOG INPUT	DIGITAL OUTPUT MSB ... LSB
$V_{IN} \geq$ Full Scale	1 ...111... 1
= Full Scale – 1 LSB	1 ...111... 1
= 1 LSB	0 ...000... 1
≤ 0	0 ...000... 0

Two's complement coding can be generated by inverting the Most Significant Bit (MSB) signal.

Adjustment Procedure – The test circuit diagram shows optional circuits for trimming the zero location and full scale gain. Because the digital outputs remain constant outside of the normal operating range (i.e. below zero and above full scale), it is recommended that transition points be used in setting the zero and full scale values. Recommended procedure is as follows:

1. Set the initiate conversion control high to provide free-run operation and verify that converter is operating.
2. Set V_{IN} to +½ LSB and trim the zero adjust circuit to obtain a 000 . . . 000 . . . to 000 . . . 001 transition. This will correctly locate the zero end.
3. For full scale adjustment, set V_{IN} to the full scale value less 1½ LSB and trim the gain adjust circuit for a 111 . . . 110 to 111 . . . 111 transition.

If adjustments are performed in this order, there should be no interaction and they should not have to be repeated.

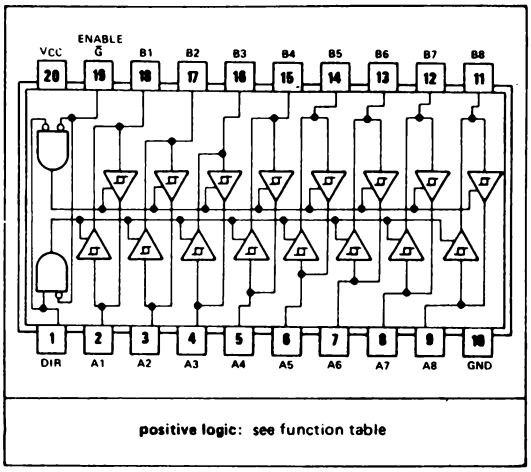
faster and the supply current will be higher. (For example: When R_{BIAS} is 20K the conversion time is reduced by 1/3, and the supply current will increase from 2mA to 7mA.) Likewise, if the R_{BIAS} is increased the conversion time will be longer and the supply current will be much lower. (For example: When R_{BIAS} = 1mΩ the conversion time will be six times longer, and the supply current is now reduced to .5 mA.) For details of this relationship refer to AN-9 typical performance curves.

FIG. 19 - Descrizione funzionale dell'8700

- **Bi-directional Bus Transceiver in a High-Density 20-Pin Package**
- **3-State Outputs Drive Bus Lines Directly**
- **P-N-P Inputs Reduce D-C Loading on Bus Lines**
- **Hysteresis at Bus Inputs Improve Noise Margins**
- **Typical Propagation Delay Times, Port-to-Port . . . 12 ns**
- **Typical Enable/Disable Times . . . 17 ns**

TYPE	I_{OL} (SINK CURRENT)	I_{OH} (SOURCE CURRENT)
SN54LS245	12 mA	−12 mA
SN74LS245	24 mA	−15 mA

SN54LS245 . . . J PACKAGE
SN74LS245 . . . J OR N PACKAGE
(TOP VIEW)



positive logic: see function table

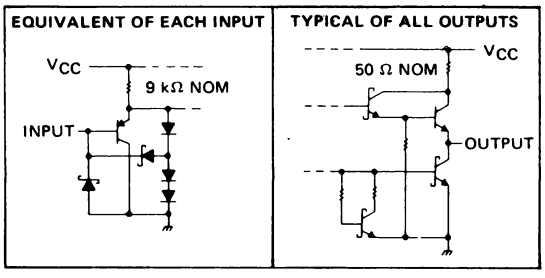
description

These octal bus transceivers are designed for asynchronous two-way communication between data buses. The control function implementation minimizes external timing requirements.

The device allows data transmission from the A bus to the B bus or from the B bus to the A bus depending upon the logic level at the direction control (DIR) input. The enable input ($\overline{G}$) can be used to disable the device so that the buses are effectively isolated.

The SN54LS245 is characterized for operation over the full military temperature range of −55°C to 125°C. The SN74LS245 is characterized for operation from 0°C to 70°C.

schematics of inputs and outputs



FUNCTION TABLE

ENABLE $\overline{G}$	DIRECTION CONTROL DIR	OPERATION
L	L	B data to A bus
L	H	A data to B bus
H	X	Isolation

H = high level, L = low level, X = irrelevant

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1) . . . . . 7 V
Input voltage . . . . . 7 V
Operating free-air temperature range: SN54LS245 . . . . . −55°C to 125°C
SN74LS245 . . . . . 0°C to 70°C
Storage temperature range . . . . . −65°C to 150°C

NOTE 1: Voltage values are with respect to network ground terminal.

FIG. 20. - Piedinatura dell'SN 74LS245 (Texas).

Una volta completata la conversione A/D l'uscita dato valido ritorna ad essere alta (fig. 18) portando l'uscita Q del flip-flop A al livello logico alto (l'ingresso D è mantenuto costantemente ad 1) e generando così una richiesta d'interruzione. La CPU a questo punto esegue la routine dell'interrupt e mediante l'istruzione IN A, (FF) abilita il tri-state, abilita il 74LS245 verso destra, riporta bassa l'uscita Q del flip-flop A e quindi alta la linea d'interruzione, e infine legge il dato convertito all'uscita del tri-state ($\overline{G_1} = \overline{G_2} = \overline{E} = 0$, DIR $= 1$) mantenendo basso l'ingresso di initiate conversion dell'8700.

Dopo aver letto e salvato in un registro o in una locazione di memoria il dato letto, il microprocessore può inviare un altro segnale all'ingresso di initiate conversion per avviare una nuova conversione.

Sono infine qui di seguito riportare le descrizioni delle istruzioni d'ingresso e d'uscita utilizzate ed il software relativo al circuito esaminato. Un circuito applicativo dell'8700 è invece mostrato in fig. 22.

Istruzione OUT (n), A

L'operando n è posto sulla parte bassa (da A_0 ad A_7) del bus degli indirizzi per selezionare il dispositivo di I/0 tra i 256 possibili. Il contenuto dell'accumulatore, inoltre è contemporaneamente presentato sulla parte alta (da A_8 ad A_{15}) del bus degli indirizzi.

Poi il byte contenuto nell'accumulatore è posto nel bus dei dati e scritto nel periferico selezionato (fig. 5).

Istruzione IN A, (n)

L'operando n è posto sulla parte bassa (da A_0 ad A_7) del bus degli indirizzi per selezionare un dispositivo di I/0 tra i 256 possibili. Contemporaneamente sulle linee da A_8 ad A_{15} del bus degli indirizzi viene posto il contenuto dell'accumulatore. In risposta a tale operazione la porta selezionata presenta sul bus dati un byte che viene così caricato nell'accumulatore del microprocessore (fig. 5).

Software:

```
Initiation   LD A, 7F
             OUT (FF), A      Inizia la conversione
             LD A, FF
             OUT (FF), A      Riporta a zero l'ingresso di initiate con-
                              version
```

```
Interrupt   PUSH AF          Salva il contenuto dei registri nello stack
            PUSH BC
            PUSH DE
            PUSH HL
            IN A, (FF)       Leggi il dato e memorizzalo nel registro B
            LD B, A
            LD A, 7F
            OUT (FF), A      Inizia un'altra conversione ed elabora il
                             dato
            LD A, FF
            OUT (FF), A
            ..............
            ..............
            ..............
            POP HL           Ripristina il contenuto dei registri
            POP DE
            POP BC
            POP AF
            RET              Ritorna al main program
```

FIG. 21. - Routine di servizio dell'interrupt.

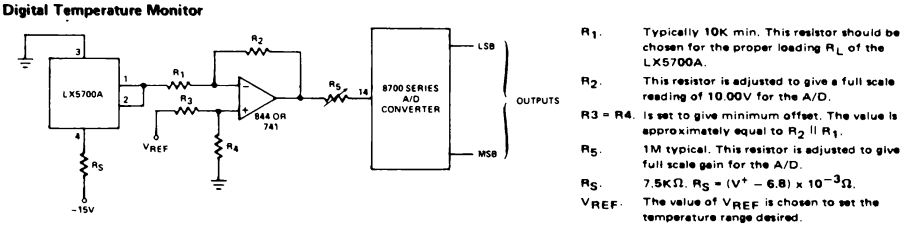


FIG. 22. - Visualizzazione digitale della temperatura.

4. Controllo di un braccio meccanico a più gradi di libertà

Il problema che viene affrontato in questo paragrafo è quello di far compiere ad un braccio meccanico una determinata sequenza di movimenti definita dall'operatore via software tramite un microcomputer. Agendo poi sul programma è possibile variare il cammino del braccio.

Spesso si definisce un braccio meccanico secondo i gradi di libertà e gli assi del movimento. In genere i sistemi di coordinate usati (fig. 23) per de-

finire la posizione nello spazio sono:
- Cartesiano: tre traslazioni lungo gli assi x, y e z (fig. a).
- Cilindrico: due traslazioni r e z e una rotazione θ (fig. b).
- Sferico: una transizione r e due rotazioni ω e φ (fig. c).
- Angolare: tre rotazioni α, β e γ (fig. d).

In questo caso il braccio robotico è del tipo a tre gradi di libertà in un sistema di riferimento cartesiano.

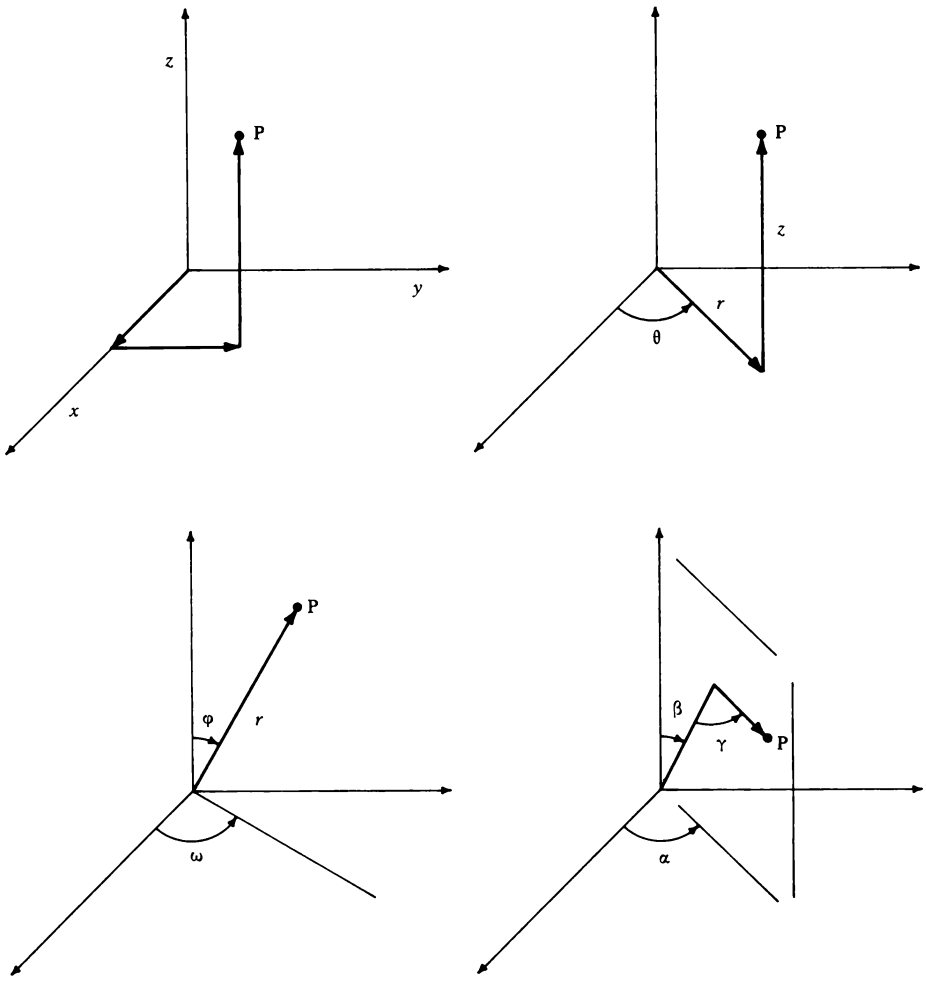


FIG. 23. - Sistemi di riferimento.

I movimenti richiesti sono realizzati mediante tre motori passo-passo che agiscono sequenzialmente lungo i tre assi cartesiani per mezzo di viti senza fine. L'interfacciamento tra i motori passo-passo ed il microcomputer avviene mediante tre schede di potenza (una per ogni motore) fornite dalla casa costruttrice del motore passo passo e un'altra progettata e realizzata dall'operatore in funzione delle specifiche richieste dal problema.

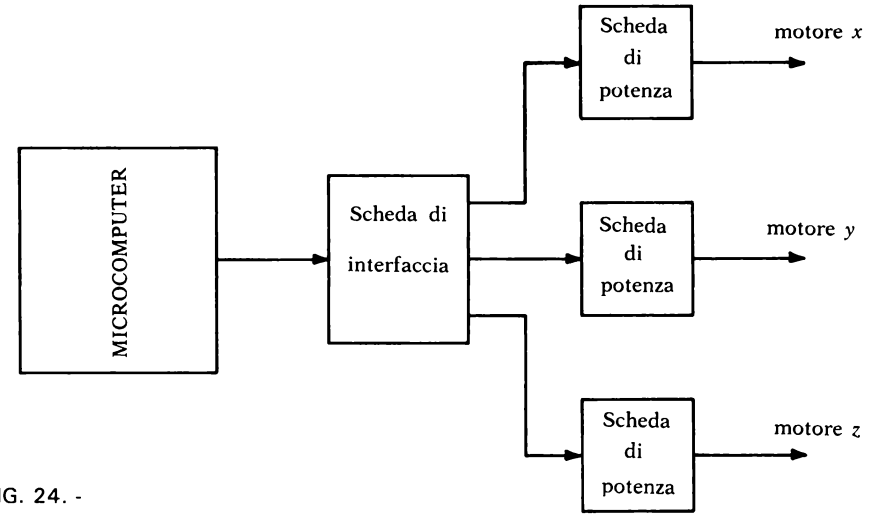


FIG. 24. -

Descrizione dei singoli blocchi:

Microcomputer: microcomputer o sistema minimo con unità centrale di elaborazione lo Z 80

Scheda di interfaccia: permette l'interfacciamento del microcomputer con le schede di potenza.

Schede di potenza: permettono il movimento dei tre motori passo-passo.

Motori passo-passo: permettono il movimento del braccio meccanico nello spazio.

Le schede di potenza, costruite dalla Sigma, e distribuite dalla Pamoco, del tipo BRL 3007, hanno le seguenti caratteristiche:
1) **Comando di passo** - il passo avviene sulla transizione alto-basso. Impedenza d'ingresso 3.3 KΩ. Livello basso $< 2V$, livello alto > 10 V.
2) **Comando di direzione** - ingresso normalmente alto che, se portato basso inverte il senso di rotazione del motore. Il segnale di direzione deve

essere valido almeno 100 μs prima del segnale di passo e deve permanervi per almeno 50 μs.

Si noti che invertendo l'eccitazione di solo una delle due fasi si inverte il senso di rotazione del motore.

3) Il 15 V uscente dal morsetto 5 non può essere utilizzato per alimentare circuiti elettronici esterni che assorbano più di 2mA.

4) Schema di cablaggio:

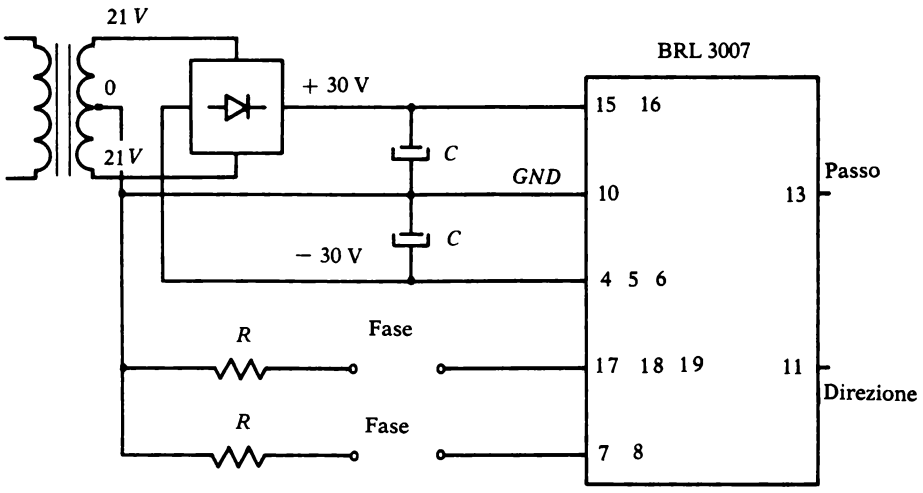


FIG. 25. - Caratteristiche della scheda di potenza.

Circuito stampato

1,2,3 - non usati
4,5,6 - -30 V_{dc}
7,8 - fase A al motore
9 - N.U.
10 - GND
11 - direzione
12 - + 15 V_{dc} (gener. inter.)
13 - comando passo
14 - N.U
15,16 - + 30 V_{dc}
17,18,19 - fase B al motore
20,21,22 - N.U.

Morsettiera

1 - fase B al motore
2,3 - + 30 V_{dc}
4 - passo
5 - + 15 V_{dc}
6 - direzione
7 - GND
8,9 - fase A al motore
10 - 30V_{dc}

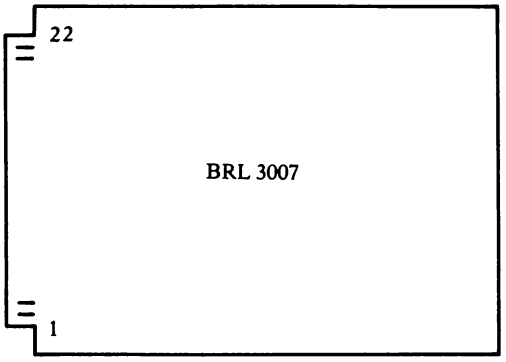


FIG. 26. - Piedinatura della scheda di potenza.

Si ricorda ora brevemente la definizione di motore passo-passo.

«Un motore passo-passo è un motore elettromagnetico in grado di convertire degli impulsi elettrici in rotazioni angolari dell'albero di uscita pari ad una frazione dell'angolo giro, che dipende dal tipo di motore, detto *passo*».

I motori utilizzati sono del tipo

20-2220 D 200-B055 (fig. 27).

In fig. 28 è mostrato lo schema completo del circuito necessario per il movimento del braccio meccanico secondo le specifiche del software riportato più avanti.

La selezione degli indirizzi dei porti di uscita è stata effettuata decodificando A_4, A_5, A_6 ed A_7 e utilizzando una tecnica di gestione di I/0 isolato. Il funzionamento del circuito è il seguente:
normalmente il demultiplexer è disabilitato e le uscite si trovano tutte allo stato alto. In queste condizioni l'uscita Q del flip-flop SR non temporizzato mantiene lo stato precedente ($S = R = 0$) ed i motori possono trovarsi posizionati in avanti o all'indietro. Il verso di ogni motore viene fissato mediante l'attivazione di Y_4 o Y_5, a seconda che lo si voglia mandare in avanti oppure all'indietro, tramite opportune istruzioni di uscita.

Così per esempio, se viene eseguita l'istruzione OUT (40),A (si tenga presente la tabella della verità dell'SN 74154 di fig. 29) le linee A_7, A_6, A_5 ed A_4 del bus degli indirizzi assumono rispettivamente lo stato 0100 (mentre le altre tre non utilizzate della parte bassa degli indirizzi sono portate tutte a zero), il demultiplexer viene abilitato ($\overline{IORQ} = \overline{WR} = 0$) ed è selezionata l'uscita Y_4. L'uscita del flip-flop si porta al livello logico 1 ($S = 1$, $R = 0$) ed in tale stato rimane fino all'esecuzione dell'istruzione OUT (50),A che seleziona l'uscita Y_5 (motore all'indietro). Successivamente vengono selezionati i motori mediante codici di dispositivo che attivano le

GENERAL SPECIFICATIONS AND ORDERING INFORMATION

TYPE Permanent magnet rotor

NO. OF PHASES Two (4 or 8 step switching sequence)

STEP ANGLE 1.8° or 2.5°

ANGULAR ACCURACY ±3% of one step, no load, after any number of steps.

AMBIENT OPERATING TEMPERATURE −20°C to 50°C without heat sink.

MAXIMUM CASE TEMPERATURE 100°C

INSULATION NEMA Class B

INSULATION RESISTANCE 1,000MΩ@ 500VDC @ 25°C

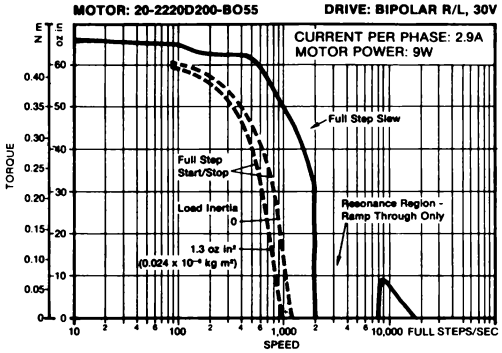
MODEL NUMBER CODE

All model numbers are complete in the table below. The following typical model number shows the codes that have been assigned to each section of the number.

20 — 2215 D 200 — B 075

- **20** — Series (rotor style)
- **2215** — diameter (2.2 inches) length (1.5 inches)
- **D** — stepping motor
- **200** — No. of steps per revolution (200 = 1.8° per step, 144 = 2.5° per step)
- **B** — No. of leads (B = 4, E = 6)
- **075** — Nominal phase resistance (0.75 ohms)

	MOTOR	NUMBER OF LEADS	TORQUE @ 50 sps oz in./Nm	HOLDING TORQUE oz in./Nm	DETENT TORQUE oz in./Nm	PHASE CURRENT Amps Unipolar	PHASE CURRENT Amps Bipolar	PHASE RESISTANCE Ohms	PHASE INDUCTANCE mH	ROTOR INERTIA oz in²/10⁻³kg m²	WEIGHT lbs/kg	PERFORMANCE CURVES
7	20-2220D200-B055	4	68/0.48	83/0.59	3.0/0.02		2.9	0.55	2.00	0.64/0.012	1.2/0.54	2-9, 2-10



1.8° and 2.5° per step
Up to 200 oz in

- Highest output torque per frame size
- Speeds to 30,000 steps/second
- Coolest running (Series 21)
- Choice of rotor styles
- Choice of driver styles

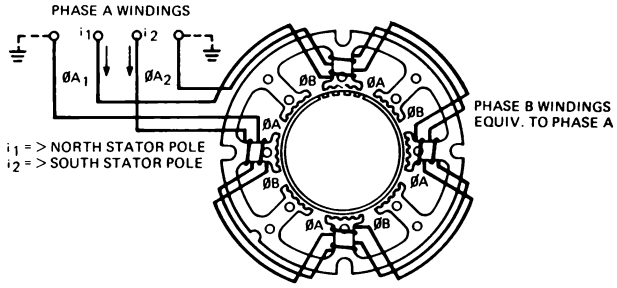


FIG. 27. - Caratteristiche del motore passo-passo.

FIG. 28. - Circuito di interfaccia per i tre motori passo-passo.

logic

FUNCTION TABLE

INPUTS						OUTPUTS													
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = high level, L = low level, X = irrelevant

FIG. 29. - Tabella della verità dell'SN 74154 (Texas)

uscite Y_1, Y_2 ed Y_3 del demultiplexer sempre tenendo presente la direzione che i motori debbono assumere di volta in volta.

Il passo del motore avviene durante la transizione dal livello logico alto a quello basso dell'uscita corrispondente del demultiplexer.

Progetto della scheda di interfaccia

Dalle note caratteristiche che descrivono la scheda di potenza BRL 3007, precedentemente riportate, si legge tra l'altro che il segnale di direzione deve essere valido almeno 100 μs prima del segnale di passo e che deve permanervi per almeno 50 μs (fig. 30).

Queste condizioni saranno soddisfatte per via software mediante l'introduzione di loop di delay opportunamente calcolati.

L'uso degli integrati 555 (Signetics) è dovuto alla necessità di allargare la durata dell'impulso di uscita del microcomputer ai 50 μs necessari per poter azionare i motori (dalla fig. 5 si può osservare che $\overline{IORQ}$ e $\overline{WR}$ restano bassi per tre periodi di clock, e quindi supposta una frequenza di lavoro di 2,5 MHz, l'impulso di OUT ha una durata di 1,2 μs).

Tenendo presente che la durata dell'impulso per il 555 usato come monostabile è data dalla relazione $t_W = 1{,}1\ R_A C$ (fig. 32), fissando il valore di R_A a 10 KΩ si ha che:

$$C = \frac{50 \cdot 10^{-6}}{1{,}1 \cdot 10^{3}} = 0{,}047\ \mu F$$

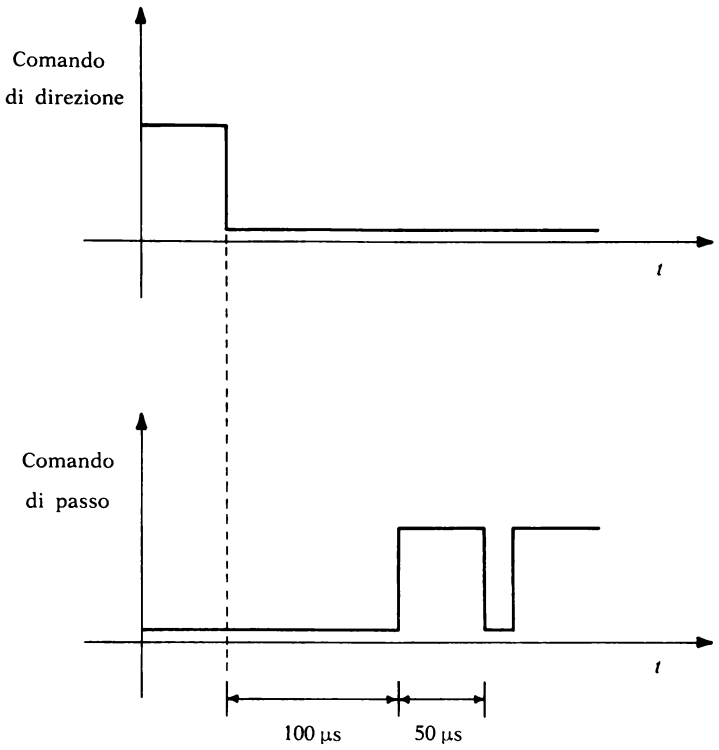


FIG. 30.

I transistori BC 141 hanno la funzione di innalzare l'ampiezza del segnale di passo ad un livello maggiore di 10 volt richiesto dal motore passo-passo per il suo funzionamento.

Il calcolo delle resistenze di collettore e di base dei transistori è stato effettuato tenendo presente che la resistenza d'ingresso della scheda di potenza è di 3,3 KΩ (fig. 31).

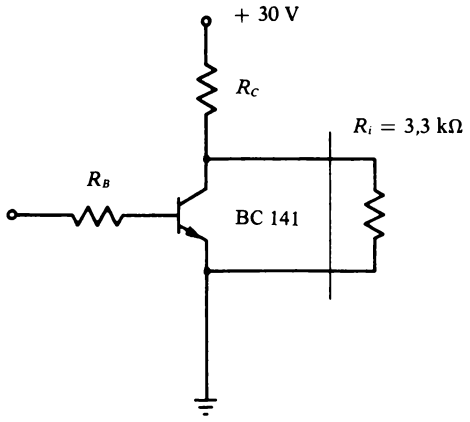


FIG. 31.

signetics

TIMER | 555

LINEAR INTEGRATED CIRCUITS

DESCRIPTION

The NE/SE 555 monolithic timing circuit is a highly stable controller capable of producing accurate time delays, or oscillation. Additional terminals are provided for triggering or resetting if desired. In the time delay mode of operation, the time is precisely controlled by one external resistor and capacitor. For a stable operation as an oscillator, the free running frequency and the duty cycle are both accurately controlled with two external resistors and one capacitor. The circuit may be triggered and reset on falling waveforms, and the output structure can source or sink up to 200mA or drive TTL circuits.

FEATURES

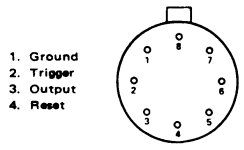
- TIMING THROUGH NINE DECADES
- OPERATES IN BOTH ASTABLE AND MONOSTABLE MODES
- ADJUSTABLE DUTY CYCLE
- HIGH CURRENT OUTPUT CAN SOURCE OR SINK 200mA
- OUTPUT CAN DRIVE TTL
- TEMPERATURE STABILITY OF 0.05% PER °C
- NORMALLY ON AND NORMALLY OFF OUTPUT

APPLICATIONS

PRECISION TIMING
PULSE GENERATION
SEQUENTIAL TIMING
TIME DELAY GENERATION
PULSE WIDTH MODULATION
PULSE POSITION MODULATION
MISSING PULSE DETECTOR

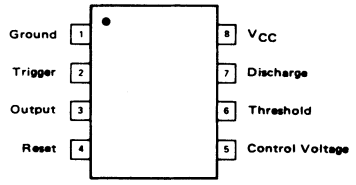
PIN CONFIGURATIONS (Top View)

T PACKAGE



ORDER PART NOS SE555T/NE555T

V PACKAGE



ORDER PART NOS. SE555V/NE555V

ABSOLUTE MAXIMUM RATINGS

Supply Voltage	+18V
Power Dissipation	600 mW
Operating Temperature Range	
NE555	0°C to +70°C
SE555	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 60 seconds)	+300°C

BLOCK DIAGRAM

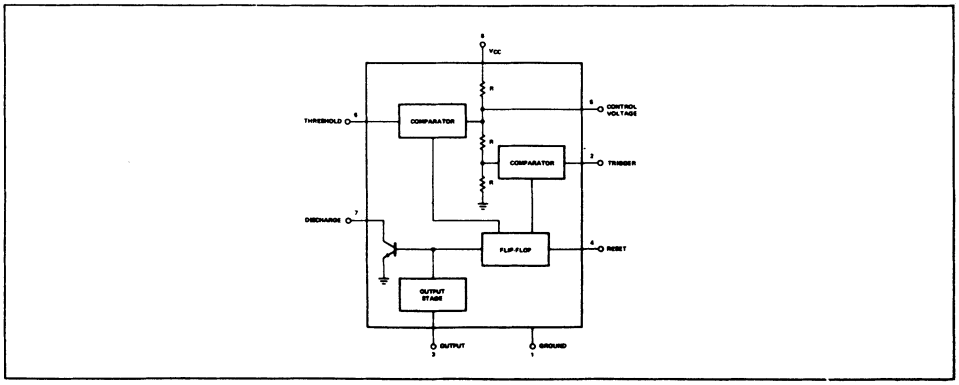


FIG. 32. - Piedinatura del 555.

ELECTRICAL CHARACTERISTICS $T_A = 25°C$, $V_{CC} = +5V$ to $+15$ unless otherwise specified

PARAMETER	TEST CONDITIONS	SE 555			NE 555			UNITS
		MIN	TYP	MAX	MIN	TYP	MAX	
Supply Voltage		4.5		18	4.5		16	V
Supply Current	$V_{CC} = 5V$ $R_L = \infty$		3	5		3	6	mA
	$V_{CC} = 15V$ $R_L = \infty$		10	12		10	15	mA
	Low State, Note 1							
Timing Error	R_A, = 1KΩ to 100KΩ							
Initial Accuracy	C = 0.1 μF Note 2		0.5	2		1		%
Drift with Temperature	see Fig. 1a $V_{CC} = 15V$		30	100		50		ppm/°C
Drift with Supply Voltage			0.05	0.2		0.1		%/Volt
Threshold Voltage			2/3			2/3		X V_{CC}
Trigger Voltage	$V_{CC} = 15V$	4.8	5	5.2		5		V
	$V_{CC} = 5V$	1.45	1.67	1.9		1.67		V
Trigger Current			0.5			0.5		μA
Reset Voltage		0.4	0.7	1.0	0.4	0.7	1.0	V
Reset Current			0.1			0.1		mA
Threshold Current	Note 3		0.1	.25		0.1	.25	μA
Control Voltage Level	$V_{CC} = 15V$	9.6	10	10.4	9.0	10	11	V
	$V_{CC} = 5V$	2.9	3.33	3.8	2.6	3.33	4	V
Output Voltage Drop (low)	$V_{CC} = 15V$							
	$I_{SINK} = 10mA$		0.1	0.15		0.1	.25	V
	$I_{SINK} = 50mA$		0.4	0.5		0.4	.75	V
	$I_{SINK} = 100mA$		2.0	2.2		2.0	2.5	V
	$I_{SINK} = 200mA$		2.5			2.5		
	$V_{CC} = 5V$							
	$I_{SINK} = 8mA$		0.1	0.25				V
	$I_{SINK} = 5mA$					.25	.35	
Output Voltage Drop (high)								
	$I_{SOURCE} = 200mA$ $V_{CC} = 15V$		12.5			12.5		
	$I_{SOURCE} = 100mA$							
	$V_{CC} = 15V$	13.0	13.3		12.75	13.3		V
	$V_{CC} = 5V$	3.0	3.3		2.75	3.3		V
Rise Time of Output			100			100		nsec
Fall Time of Output			100			100		nsec

NOTES

1. Supply Current when output high typically 1mA less.
2. Tested at $V_{CC} = 5V$ and $V_{CC} = 15V$
3. This will determine the maximum value of $R_A + R_B$. For 15V operation, the max total R = 20 megohm.

EQUIVALENT CIRCUIT (Shown for One Side Only)

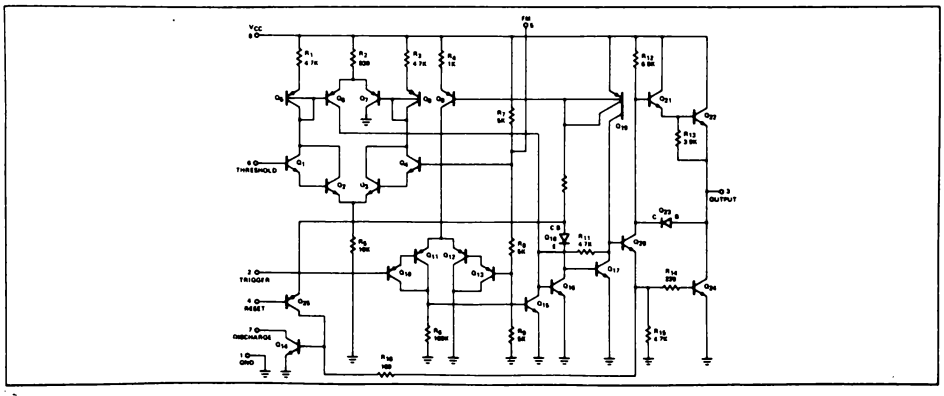


FIG. 33. - Caratteristiche elettriche del 555.

APPLICATIONS INFORMATION

MONOSTABLE OPERATION

In this mode of operation, the timer functions as a one-shot. Referring to Figure 1a the external capacitor is initially held discharged by a transistor inside the timer.

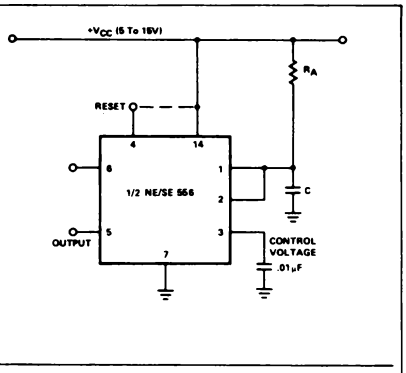


FIGURE 1a.

Upon application of a negative trigger pulse to pin 2, the flip-flop is set which releases the short circuit across the external capacitor and drives the output high. The voltage across the capacitor, now, increases exponentially with the time constant $\tau = R_A C$. When the voltage across the capacitor equals 2/3 V_{CC}, the comparator resets the flip-flop which in turn discharges the capacitor rapidly and drives the output to its low state. Figure 1b shows the actual waveforms generated in this mode of operation.

The circuit triggers on a negative going input signal when the level reaches 1/3 V_{CC}. Once triggered, the circuit will remain in this state until the set time is elapsed, even if it is triggered again during this interval. The time that the output is in the high state is given by $t = 1.1\ R_A C$ and can easily be determined by Figure 1c. Notice that since the charge rate, and the threshold level of the comparator are both directly proportional to supply voltage, the timing

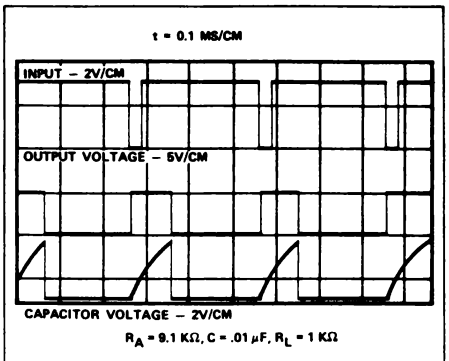


FIGURE 1b.

interval is independent of supply. Applying a negative pulse simultaneously to the reset terminal (pin 4) and the trigger terminal (pin 2) during the timing cycle discharges the external capacitor and causes the cycle to start over again. The timing cycle will now commence on the positive edge of the reset pulse. During the time the reset pulse is applied, the output is driven to its low state.

When the reset function is not in use, it is recommended that it be connected to V_{CC} to avoid any possibility of false triggering.

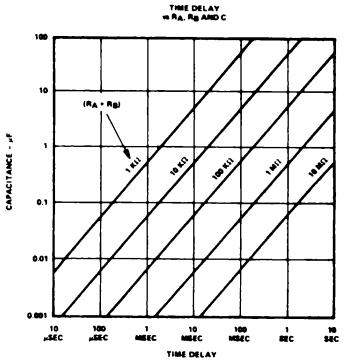


FIGURE 1c.

ASTABLE OPERATION

If the circuit is connected as shown in Figure 2a (pins 2 and 6 connected) it will trigger itself and free run as a multivibrator. The external capacitor charges through R_A and R_B and discharges through R_B only. Thus the duty cycle may be precisely set by the ratio of these two resistors.

FIG. 34. - 555 usato come monostabile.

SIGNETICS TIMERS ■ 555/556

APPLICATIONS INFORMATION (Cont'd)

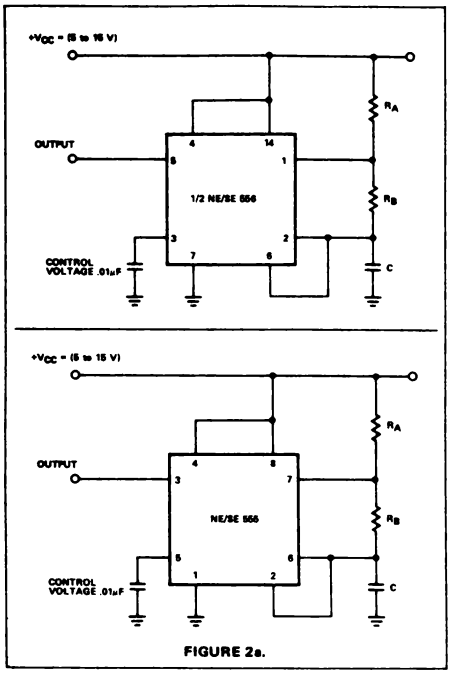


FIGURE 2a.

In this mode of operation, the capacitor charges and discharges between 1/3 V_{CC} and 2/3 V_{CC}. As in the triggered mode, the charge and discharge times, and therefore the frequency are independent of the supply voltage.

Figure 2b shows actual waveforms generated in this mode of operation.

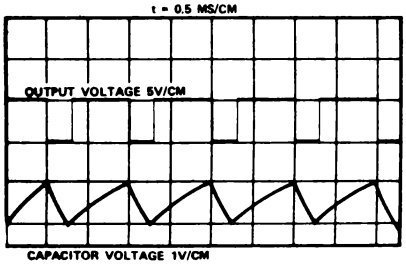


FIGURE 2b.

The charge time (output high) is given by:

$$t_1 = 0.693 (R_A + R_B) C$$

and the discharge time (output low) by:

$$t_2 = 0.693 (R_B) C$$

Thus the total period is given by:

$$T = t_1 + t_2 = 0.693 (R_A + 2R_B) C$$

The frequency of oscillation is then:

$$f = \frac{1}{T} = \frac{1.44}{(R_A + 2R_B) C}$$

and may be easily found by Figure 2c.

The duty cycle is given by:

$$D = \frac{R_B}{R_A + 2R_B}$$

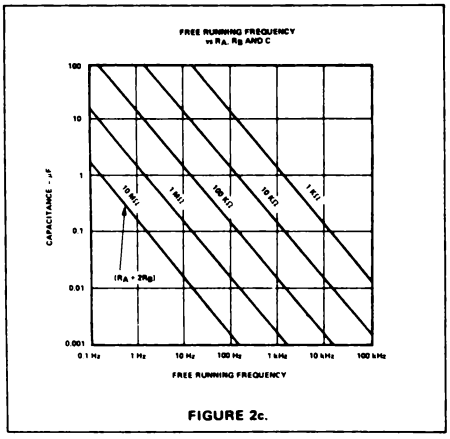
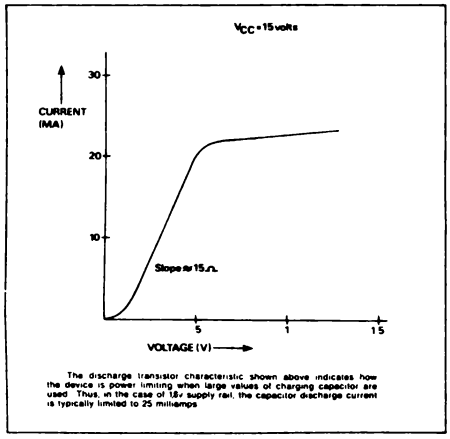


FIGURE 2c.



The discharge transistor characteristic shown above indicates how the device is power limiting when large values of charging capacitor are used. Thus, in the case of 1.8v supply rail, the capacitor discharge current is typically limited to 25 milliamps

FIG. 35. - 555 usato come astabile.

Quando il transistore è ON (ingresso al livello logico alto) la tensione di collettore vale circa 0V, mentre quando il transistore è OFF (ingresso al livello logico basso) la tensione di collettore vale circa 15 volt ($R_C = R_i = 3{,}3$ K Ω), valore sufficiente quest'ultimo per azionare il motore durante le commutazioni.

Dalla consultazione delle caratteristiche del BC141 si legge tra l'altro che:

$V_{CEOmax} = 100$ V $\qquad h_{FEMin} = 40$

$I_{COMax} = 1$ A

cosicchè:

$$I_{Csat} \cong \frac{V_{cc}}{R_C} = \frac{30}{3{,}3 \cdot 10^3} = 9 \text{ mA}$$

dovendo essere:

$$\frac{I_C}{I_B} \leqq h_{FEMin}$$

ponendo

$$\frac{I_C}{I_B} = 20$$

si ha che $I_B = 0{,}45$ mA

essendo poi

$V_i = R_B I_B + V_{BEsat}$ $\qquad$ con $V_i = 5$V

si ha

$$R_B = \frac{5 - 0{,}8}{0{,}45 \cdot 10^{-3}} = 10 \text{ K } \Omega$$

I resistori da 120 Ω, 20 W, infine, hanno la funzione di limitare la corrente assorbita dalle due fasi dei motori ad un valore che rientra nel campo della loro applicazione senza danneggiarli.

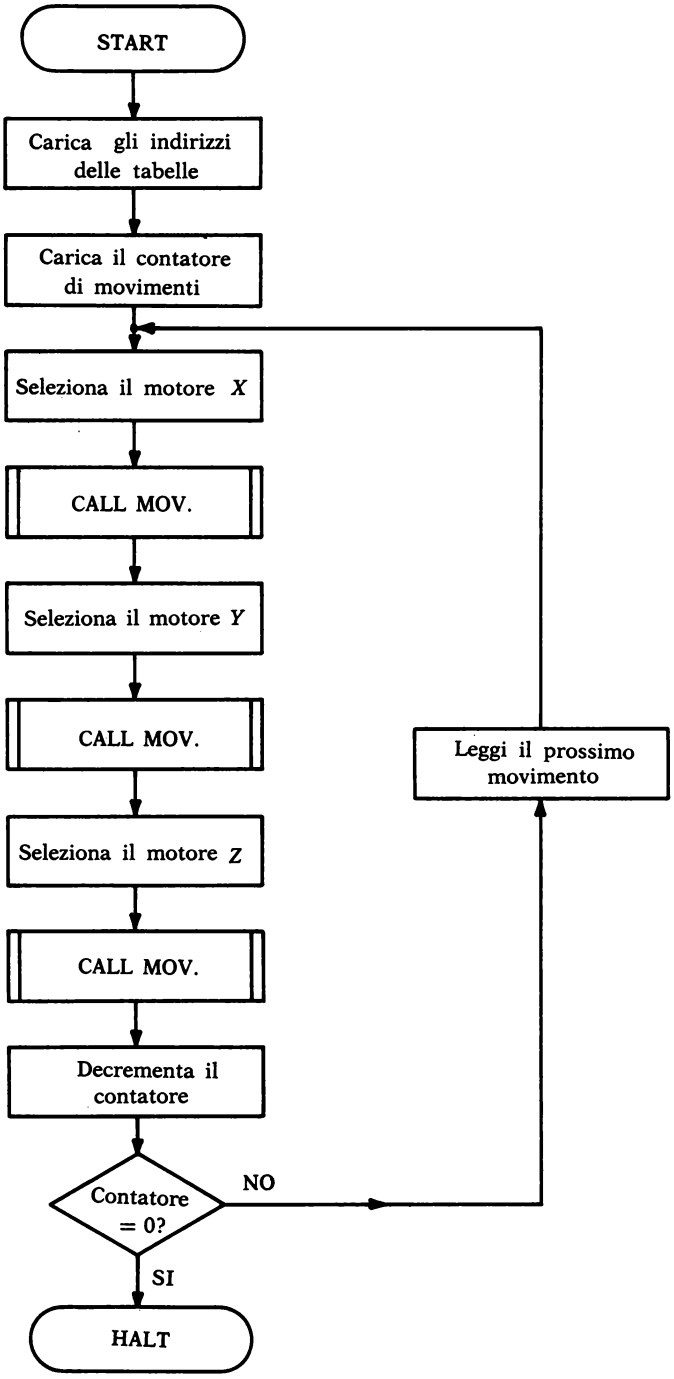


FIG. 36. - Schema a blocchi del programma principale.

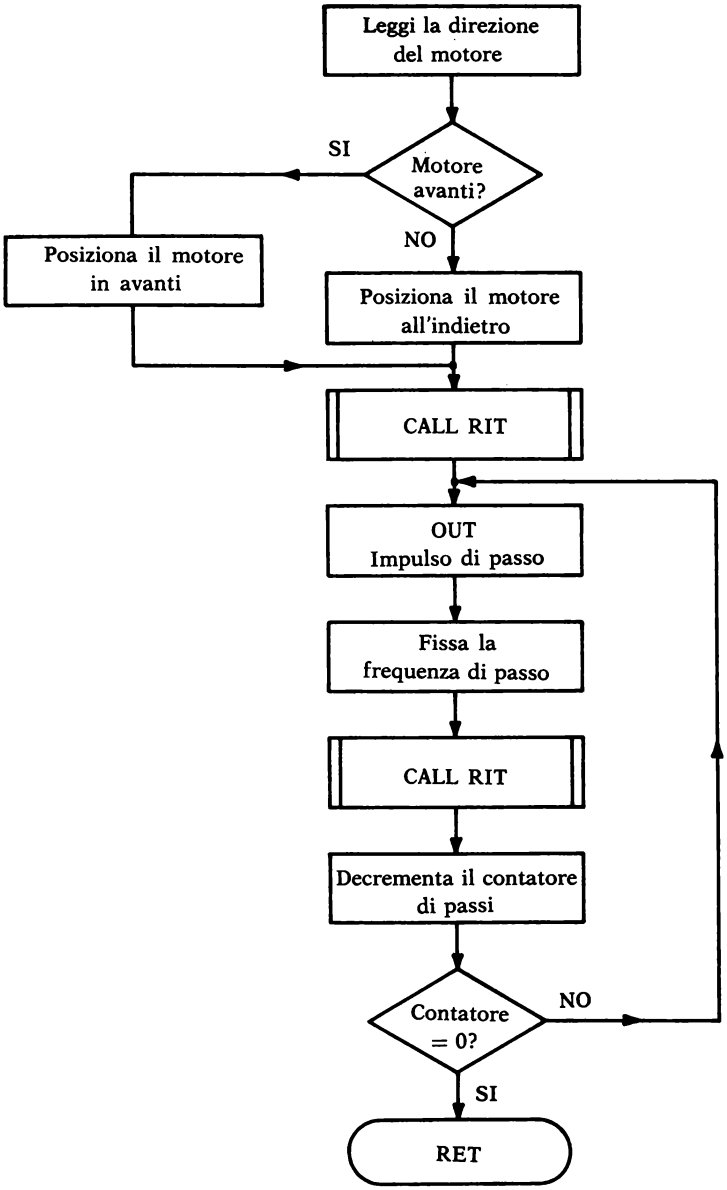


FIG. 37. - Schema a blocchi della subroutine «Movimento».

Software

Il seguente programma, il cui flow-chart è riportato nelle fig. 36, 37 e 38 realizza il movimento dei tre motori passo-passo secondo una sequenza di movimenti le cui entità, scritte in passi, sono memorizzate in tre tabelle. Il numero dei movimenti per ogni motore in questo caso è uguale a sette, alla fine dei movimenti il braccio meccanico ritorna automaticamente alla posizione di partenza.

Resta inteso che variando i contenuti delle tabelle ed il contenuto del registro contatore di movimenti è possibile far effettuare al braccio meccanico tracciati più o meno complessi.

Caricamento del numero dei passi nelle tre tabelle:

TAB X: +50, +100, +127, -30, -40, -100, -107
TAB Y: +30, -20, +100, +70, -80, +20, -120
TAB Z: +40, +70, +20, -127, +40, +60, -103

Il segno positivo o negativo prima del numero dei passi indica se il movimento del motore deve essere effettuato in avanti o all'indietro.

Gli indirizzi delle tre tabelle sono stati assunti uguali a 0100, 0110, 0120.

Il numero dei passi tradotto in esadecimale, tenendo conto dei segni (cap. 2 parag. 4), risulta:

TAB X (0100): 32, 64, 7F, E2, D8, 9C, 95
TAB Y (0110): 16, EC, 64, 46, B0, 14, 88
TAB Z (0120): E2, 46, 14, 81, E2, 3C, 99

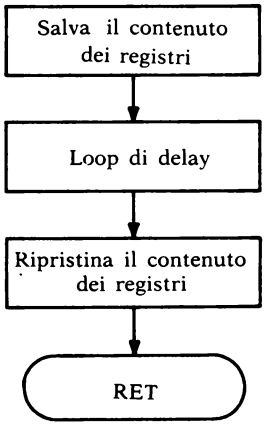


FIG. 38. - Schema a blocchi della subroutine «Ritardo».

Il programma supposto caricato a partire dalla locazione di memoria 0200 è il seguente:

Main Program:

	0200	31 00 08	LD SP, 0800	Definisci l'area di stack
	0203	06 07	LD B, 07	Carica il numero dei movimenti
	0205	00	NOP	
	0206	21 20 01	LD HL, 0120	Carica l'indirizzo della tabella Z
	0209	FD 21 10 01	LD IY, 0110	Carica l'indirizzo della tabella Y
	020D	DD 21 00 01	LD IX, 0100	Carica l'indirizzo della tabella X
P_0	0211	DD 56 00	LD D, (IX + d)	Carica il contenuto del registro IX sommato allo spostamento d (inizialmente uguale a 00) in D
	0214	0E 10	LD C,10	Carica il codice dispositivo per il motore X
	0216	CD 00 03	CALL MOV	
	0219	FD 56 00	LD D, (IY + d)	
	021C	0E 20	LC C, 20	Carica il codice dispositivo per il motore Y
	021E	CD 00 03	CALL MOV	
	0221	56	LD D, (HL)	
	0222	0E 30	LD C, 30	Carica il codice dispositivo per il motore Z
	0224	CD 00 03	CALL MOV	
	0227	05	DEC B	Decrementa il contatore di movimenti
	0228	C2 2C 02	JP NZ, P_1	Salta a P_1 se il contatore è diverso da zero
	022B	76	HALT	
P_1	022C	DD 23	INC IX	Leggi il prossimo movimento
	022E	FD 23	INC IY	
	0230	23	INC HL	
	0231	C3 11 02	JP P_0	

Subroutine MOV

	0300	7A	LD A,D	
	0301	A7	AND A	Testa il flag di segno
	0302	F2 0D 03	JP P, P_2	Salta a P_2 se il flag di segno è positivo
	0305	D3 50	OUT (50), A	Posiziona il motore all'indietro
	0307	ED 44	NEG (*)	Complementa a due l'accumulatore
	0309	57	LD D,A	
	030A	C3 0F 03	JP, P_3	
P_2	030D	D3 40	OUT (40), A	Posiziona il motore in avanti
P_3	030F	CD 00 04	CALL RIT	
P_4	0312	ED 79	OUT (C), A	Genera un impulso di passo
	0314	1E 19	LD E,19 (**)	Regola la velocità di passo
P_5	0316	CD 00 04	CALL RIT	
	0319	00	NOP	
	031A	1D	DEC E	
	031B	00	NOP (***)	

	031C	C2 16 03	JP NZ, P_5	
	031F	15	DEC D	Decrementa il numero dei passi
	0320	00	NOP	
	0321	C2 12 03	JP NZ, P_4	
	0324	C9	RET	Ritorna al main program
Subroutine RIT				
	0400	C5	PUSH BC	Salva il contenuto dei registri B e C
	0401	06 1C	LD B, 1C	Loop di ritardo necessario per il posizionamento della direzione dei motori
P_6	0403	05	DEC B	
	0404	C2 03 04	JP NZ, P_6	
	0407	C1	POP BC	Ripristina il contenuto dei registri
	0408	C9	RET	

In fig. 39 viene infine riportato il diagramma temporale riepilogativo di movimento del sistema considerato relativo al motore passo-passo agente sull'asse X. Come si può notare dalla figura, se non si vuole la sovrapposizione degli impulsi di passo (blocco del motore), occorre scegliere una frequenza di rotazione minore di 20 KH_Z (valore largamente superiore a quello ammesso del motore che è di circa 500 Hz). Chiaramente analoghe considerazioni possono essere fatte per gli altri due motori agenti sull'asse Y e Z.

Note

(*) L'istruzione NEG è necessaria in quanto supponendo, per esempio, caricato nel registro D - 30 (11100010) il motore sarebbe posizionato all'indietro (il bit più significativo vale 1) ma il numero dei passi che esso effettuerebbe non sarebbe uguale a 30 bensì

$$(11100010)_2 = (226)_{10}$$

e quindi farebbe 226 passi all'indietro anzichè i 30 previsti.

Con l'istruzione NEG l'accumulatore, che contiene il dato del registro D, viene complementato a due:

$$(00011110)_2 = (1E)_{16} = (30)_{10}$$

ed il numero dei passi eseguito è quello voluto.

(**) Può essere facilmente dimostrato, calcolando i tempi di esecuzione delle istruzioni, che il dato esadecimale 19 permette il funzionamento dei motori alla massima frequenza ammissibile senza che questi si blocchino.

La minima velocità di passo dei motori si ottiene caricando nel registro E l'esadecimale 01.

(***) Le istruzioni di NOP presenti nel programma non hanno nessun significato particolare essendo state introdotte per correggere degli indirizzi senza per questo riscrivere l'intero programma.

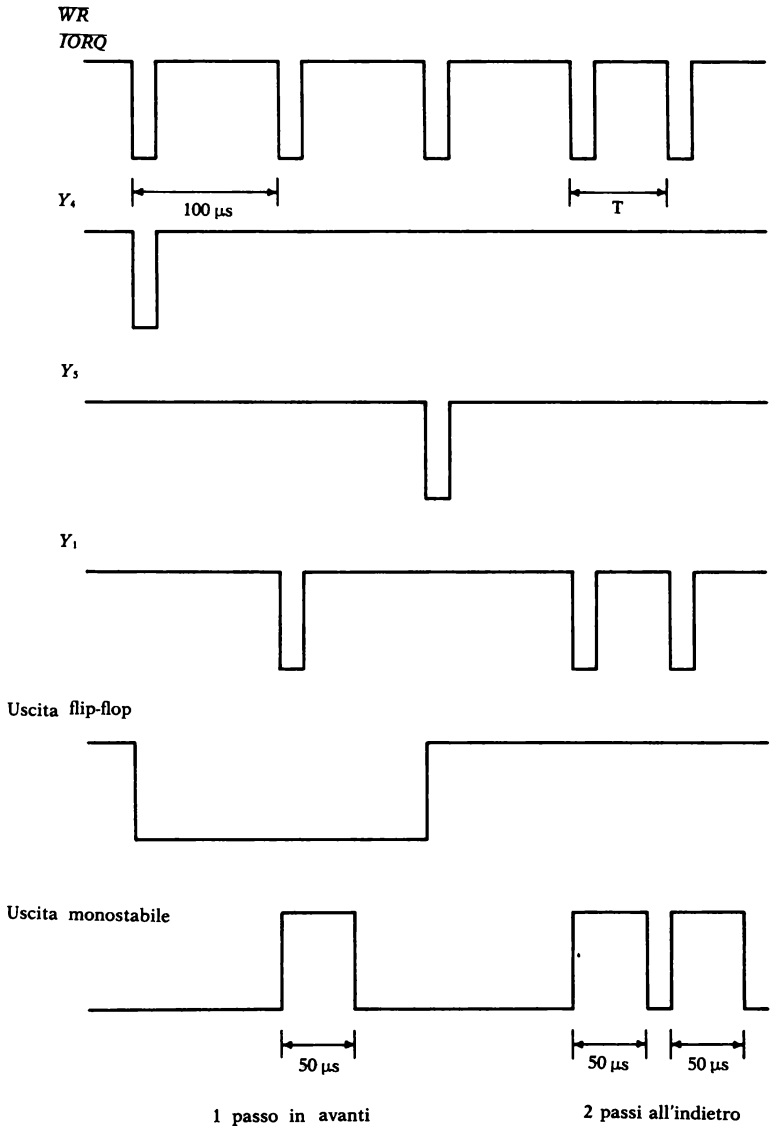


FIG. 39. - Diagramma temporale relativo al motore passo-passo agente lungo l'asse X.

5. Conversione Analogico/Digitale

Nel Volume Elettronica Digitale si è visto come può essere effettuata una conversione analogico-digitale (A/D) esclusivamente per via hardware utilizzando integrati dedicati quali i convertitori A/D.

In questo paragrafo è mostrato un semplice esempio di conversione A/D eseguito essenzialmente per via software mediante un sistema a microprocessore. Lo schema è quello di fig. 40 in cui un timer 555 in configurazione astabile fornisce un segnale impulsivo alla CPU Z 80 tramite un PI0 Z 80.

La frequenza del segnale è regolabile per mezzo di un potenziometro. La porta A del PI0 è definita come porta di uscita mentre la porta B come porta di ingresso.

Il software del sistema esegue un loop a velocità costante campionando l'uscita del timer collegata all'ingresso B_0 del PI0. Un contatore (coppia di registri DE) tiene conto del numero dei loop eseguiti durante il tempo in cui è alta l'uscita del 555.

Variando la resistenza del potenziometro varia la frequenza del timer e

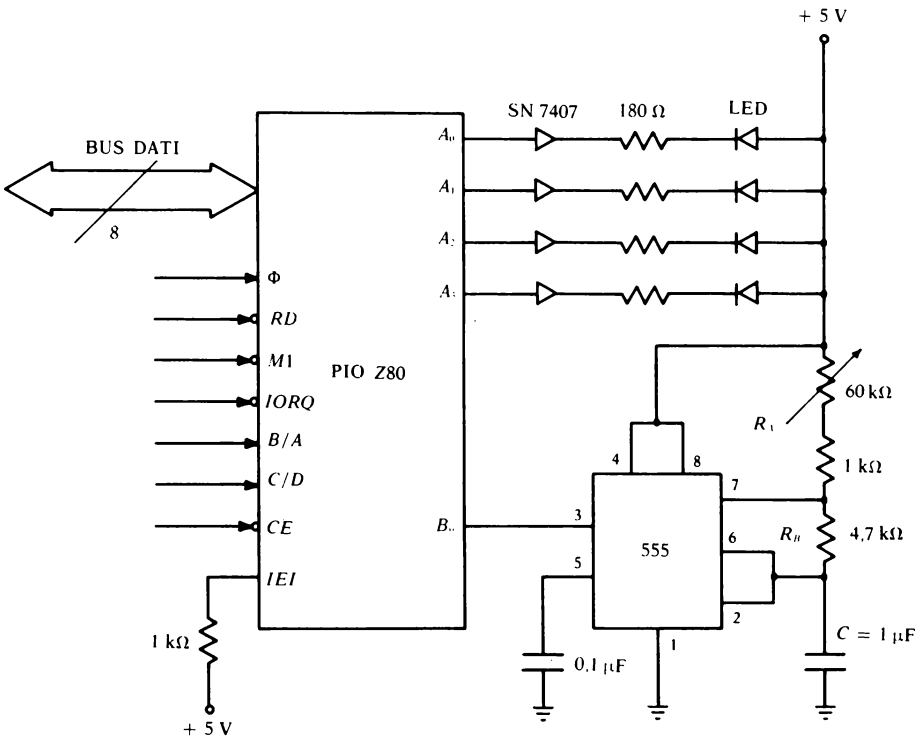


FIG. 40. - Conversione Analogico-Digitale.

quindi anche la configurazione d'uscita visualizzata sotto forma digitale dei quattro diodi led.

I valori delle resistenze e della capacità sono stati calcolati in modo da poter variare le configurazioni di uscita da 0000 a 1111 secondo la tabella 1.

Dall'esame della tabella 1 si può notare che il passaggio da una configurazione d'uscita all'altra non avviene con uno scarto di frequenza costante; ciò è dovuto alla non linearità della relazione della frequenza di oscillazione del timer in funzione della variazione di R_A.

Dalla fig. 35 si ha infatti che la frequenza di oscillazione del 555 vale:

$$f = \frac{1{,}44}{(R_A + 2R_B) \cdot C}$$

La tolleranza dei componenti può richiedere un aggiustamento dei loro valori in modo da assicurare un corretto incremento di conteggio.

Resistenza R_A (K Ω)	Frequenza del timer (Hz)	Uscita digitale
1	138	1111
5	100	1110
9	78	1101
13	64	1100
17	55	1011
21	47	1010
25	42	1001
29	37	1000
33	34	0111
37	31	0110
41	29	0101
45	26	0100
49	25	0011
53	23	0010
57	22	0001
61	20,4	0000

Tabella 1

La stessa conversione A/D potrebbe essere fatta per mezzo del CTC Z 80 (cap. 4° parag. 14) collegando l'uscita del 555 all'ingresso clock/trigger del canale 0 del CTC seguendo due diversi metodi.

Un metodo è quello di usare un altro canale del CTC per generare delle interruzioni ad ogni prefissato periodo di tempo. In questi istanti può essere letto il canale 0 ed il numero di transizioni può essere usato per calcolare la posizione del potenziometro.

L'altro metodo è quello di connettere anche il canale 1 all'uscita del timer. Il canale 0 può così misurare il tempo in cui gli impulsi sono alti mentre il canale 1 genera delle interruzioni quando gli impulsi vanno bassi.

Dopo la ricezione dell'interruzione la CPU sarà in grado di effettuare la lettura del 555.

Software

Indirizzi assegnati ai porti del PIO (cap. 4° parag. 7).
PIO porta A dati = 08
PIO porta A controllo = 0A
PIO porta B dati = 09
PIO porta B controllo = 0B

Programma:

	0100	31 00 08	LD SP, 0800	Inizializza lo stack pointer
	0103	3E 0F	LD A, 0F	Predisponi il modo di uscita per la porta A
	0105	D3 0A	OUT (0A), A	
	0107	3E 4F	LD A, 4F	Predisponi il modo di ingresso per la porta B
	0109	D3 0B	OUT (0B), A	
	010B	3E 07	LD A, 07	Disabilita le interruzioni sulla porta A
	010D	D3 0A	OUT (0A), A	
	010F	D3 0B	OUT (0B), A	Disabilita le interruzioni sulla porta B
P_0	0111	DB 09	IN A, (09)	Leggi l'uscita del timer
	0113	CB 47	BIT 0, A	L'uscita è alta? Se non lo è salta a P_0
	0115	20 FA	JR NZ, P_0	
	0117	11 00 00	LD E, 00	Azzera il contatore di cicli
P_1	011A	DB 09	IN A, (09)	Leggi l'uscita del timer
	011C	CB 47 (*)	BIT 0,A	L'uscita è alta ? Se non lo è salta a P_1
	011E	28 FA	JR NZ, P_1	

P_2	0120	DB 09	IN A, (09)	Leggi l'uscita del timer
	0122	13	INC DE	Incrementa il contatore di cicli
	0123	CB 47	BIT 0, A	L'uscita del timer è appena passata al livello logico basso? Se si blocca il loop.
	0125	20 F9	JR NZ, P_2	
	0127	7A	LD A,D	Carica il numero di cicli contati nell'accumulatore
	0128	D3 08	OUT (08), A	Visualizza il numero
	012A	C3 11 01	JP P_0	Effettua un nuovo ciclo di letture.

(*) L'istruzione BIT b,r testa il bit b del registro r agendo sul registro di flag nel modo seguente:
S = non definito
Z = 1 se il bit testato è zero, uguale a 0 in caso contrario
H = 1
P/V = non definito
N = 0
C = inalterato

6. Conversione Digitale/Analogica

In questo paragrafo viene affrontato lo studio della generazione di una forma d'onda sinusoidale e di una a dente di sega, quest'ultima generata mediante l'esecuzione di una subroutine d'interrupt, utilizzando un sistema a microprocessore interfacciato ad un convertitore digitale-analogico (volume Elettronica Digitale cap. 7°).

Naturalmente variando opportunamente il programma è possibile generare forme d'onda diverse. Lo schema utilizzato è quello di fig. 41 in cui il DAC (Digital Analogic Converter) viene visto dalla CPU Z 80 come porto di uscita per una tecnica di I/0 isolato effettuando per semplicità la decodifica di una sola linea della parte bassa del bus degli indirizzi, per esempio A_6.

Ogni volta che viene eseguita una istruzione di OUT in cui $A_6 = 1$, l'uscita della porta logica Or si porta al livello basso permettendo al DAC di effettuare la conversione del dato presente ai suoi ingressi. Come è possibile osservare dalle caratteristiche del DAC riportate più avanti il DAC 0830 (National) utilizzato è un convertitore ad otto bit bipolare, direttamente compatibile con il bus del microprocessore, munito di latch interno.

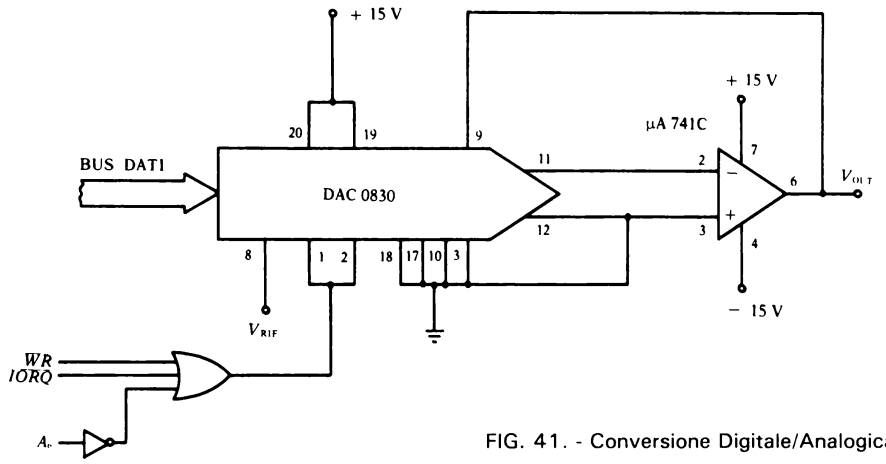


FIG. 41. - Conversione Digitale/Analogica.

La linearità delle uscite $I_{out\,1}$ ed $I_{out\,2}$ del DAC dipende dal valore della loro tensione che conviene tenere costantemente uguale a GND tramite la massa virtuale dell'amplificatore operazionale, in configurazione invertente con guadagno unitario, necessario per il corretto funzionamento del sistema. Come amplificatore operazione è stato scelto il μA 741C le cui caratteristiche sono mostrate nelle fig. 49 e 50.

In queste condizioni si ha che la tensione di uscita del μA 741 vale:

$$V_{out} = -(I_{out\,1} \cdot R_{fb}) = \frac{-V_{REF}\,(\text{Digital Input})_{10}}{256}$$

Il programma relativo alla generazione della forma d'onda sinusoidale riportato più avanti prevede la lettura di una tabella di valori del seno della funzione calcolati ogni 11,25° (fig. 48) in ordine crescente e decrescente tenendo presente che per mezzo periodo tali valori debbono essere complementati a due.

tabella

ωt	sen ωt	sen $\omega t \cdot 128$	valore esad.	valore esad. $+ 80_H$
0	0	0	00	80
11,25°	0,195	24,96	19	99
22,5	0,382	48,89	31	B1
33,75	0,555	71,04	47	C7
45	0,707	90,49	5A	DA
56,25	0,831	106,3	6A	EA
67,5	0,923	118,14	76	F6
78,75	0,980	125,4	7D	FD
90	1	128	7F	FF

National Semiconductor

DECEMBER 1980

MICRO-DAC™ DAC0830/0831/0832 8-Bit μP Compatible, Double-Buffered D to A Converters

General Description

The DAC0830 is an advanced CMOS/Si-Cr 8-bit multiplying DAC designed to interface directly with the 8080, 8048, 8085, Z-80, and other popular microprocessors. A deposited silicon-chromium R-2R resistor ladder network divides the reference current and provides the circuit with excellent temperature tracking characteristics (0.05% of Full Scale Range maximum linearity error over temperature). The circuit uses CMOS current switches and control logic to achieve low power consumption and low output leakage current errors. Special circuitry provides TTL logic input voltage level compatibility.

Double buffering allows these DACs to output a voltage corresponding to one digital word while holding the next digital word. This permits the simultaneous updating of any number of DACs.

The DAC0830 series are the 8-bit members of a family of microprocessor-compatible DAC's (MICRO-DAC's™). For applications demanding higher resolution, the DAC1000 series (10-bits) and the DAC1208 and DAC1230 (12-bits) are available alternatives.

Micro-Dac is a trademark of National Semiconductor Corp.

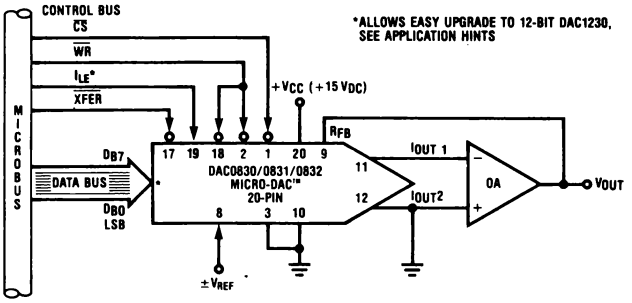
Features

- Double-buffered, single-buffered or flow-through digital data inputs
- Easy interchange and pin-compatible with 12-bit DAC1230 series
- Direct interface to all popular microprocessors
- Linearity specified with zero and full scale adjust only—NOT BEST STRAIGHT LINE FIT.
- Works with ±10V reference-full 4-quadrant multiplication
- Can be used in the voltage switching mode
- Logic inputs which meet TTL voltage level specs (1.4V logic threshold)
- Operates "STAND ALONE" (without μP) if desired

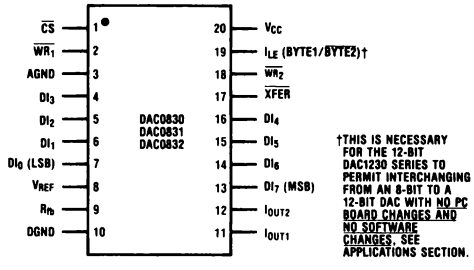
Key Specifications

■ Current settling time	1 μs
■ Resolution	8-bits
■ Linearity (guaranteed over temp.)	8, 9, or 10 bits
■ Gain Tempco	0.0002% FS/°C
■ Low power dissipation	20 mW
■ Single power supply	5 to 15 V_{DC}

Typical Application



Pin Configuration Top View



©1980 National Semiconductor Corp.

DA-FL20M120/Printed in U.S.A.

MICRO-DAC™ DAC0830/0831/0832 8-Bit μP Compatible, Double-Buffered D to A Converters

FIG. 42. - Caratteristiche e piedinatura del DAC 0830 (National).

Absolute Maximum Ratings (Notes 1 and 2)

Supply Voltage (V_{CC})	17 V_{DC}
Voltage at any digital input	V_{CC} to GND
Voltage at V_{REF} input	±25V
Storage temperature range	−65°C to +150°C
Package dissipation at T_A = 25°C (Note 3)	500 mW
DC voltage applied to I_{OUT1} or I_{OUT2} (Note 4)	−100 mV to V_{CC}
Lead temperature (soldering, 10 seconds)	300°C

Operating Ratings

Temperature Range	
Part numbers with 'LCN' suffix	0°C to 70°C
Part numbers with 'LCD' suffix	−40°C to +85°C
Part numbers with 'LD' Suffix	−55°C to +125°C
Voltage at any digital input	V_{CC} TO GND

General Electrical Characteristics T_A = 25°C, V_{REF} = 10.000 V_{DC} unless otherwise noted

Parameter	Conditions	See Note	V_{CC} = 12V_{DC} ± 5% to 15V_{DC} ± 5%			V_{CC} = 5V_{DC} ± 5%			Units
			Min.	Typ.	Max.	Min.	Typ.	Max.	
Resolution			8	8	8	8	8	8	bits
Linearity Error	Zero and full scale adjusted	4,7							
	$T_{MIN} < T_A < T_{MAX}$	6							
	$-10V \leq V_{REF} \leq +10V$	5							
	DAC0830				0.05			0.05	% of FSR
	DAC0831				0.1			0.1	% of FSR
	DAC0832				0.2			0.2	% of FSR
Differential Nonlinearity	Zero and full scale adjusted	4,7							
	$T_{MIN} < T_A < T_{MAX}$	6							
	$-10V \leq V_{REF} \leq +10V$	5							
	DAC0830				0.1			0.1	% of FSR
	DAC0831				0.2			0.2	% of FSR
	DAC0832				0.4			0.4	% of FSR
Monotonicity	$T_{MIN} < T_A < T_{MAX}$	4,6							
	$-10V \leq V_{REF} \leq +10V$	5	8	8	8	8	8	8	bits
Gain Error	Using internal R_{fb}								
	$-10V \leq V_{REF} \leq +10V$	5	−1.0	±0.2	1.0	−1.0	±0.2	1.0	% of FS
Gain Error Tempco	$T_{MIN} < T_A < T_{MAX}$	6							
	Using internal R_{fb}	10		0.0002	0.0006		0.0002	0.0006	% of FS/°C
Power Supply Rejection	All digital inputs latched high								
	V_{CC} = 14.5V to 15.5V			0.0002					% FSR/V
	11.5V to 12.5V			0.0006					% FSR/V
	4.5V to 5.5V						0.0130		%FSR/V
Reference Input Resistance			10	15	20	10	15	20	kΩ
Output Feedthrough Error	V_{REF} = 20V_{P-P}, f = 100 kHz All data inputs latched low								
	D Package	9		3			3		mV_{P-P}
	N Package			3			3		mV_{P-P}
Output Capacitance I_{OUT1}	All data inputs			70			70		pF
Capacitance I_{OUT2}	latched low			200			200		pF
I_{OUT1}	All data inputs			200			200		pF
I_{OUT2}	latched high			70			70		pF
Supply Current Drain	$T_{MIN} \leq T_A \leq T_{MAX}$	6		1.2	2.0		1.2	2.0	mA

FIG. 43. - Caratteristiche elettriche del DAC 0830.

General Electrical Characteristics $T_A = 25°C$, $V_{REF} = 10.000\,V_{DC}$ unless otherwise noted

Parameter		Conditions	See Note	$V_{CC} = 12V_{DC} \pm 5\%$ to $15V_{DC} \pm 5\%$			$V_{CC} = 5V_{DC} \pm 5\%$			Units
				Min.	Typ.	Max.	Min.	Typ.	Max.	
Output Leakage Current	I_{OUT1}	$T_{MIN} \leq T_A \leq T_{MAX}$ All data inputs latched low	6 11			100			100	nA
	I_{OUT2}	All data inputs latched high				100			100	nA
Digital Input Voltages		$T_{MIN} \leq T_A \leq T_{MAX}$ Low Level	6							
		LD suffix				0.8			0.6	V_{DC}
		Parts with LCD or LCN suffix				0.8			0.8	V_{DC}
		High Level-All Parts		2.0			2.0			V_{DC}
Digital Input Currents		$T_{MIN} \leq T_A \leq T_{MAX}$	6							
		Digital inputs < 0.8V			−50	−200		−50	−200	μA_{DC}
		Digital inputs > 2.0V			0.1	+10		0.1	+10	μA_{DC}
Current Settling Time	t_S	$V_{IL} = 0V$, $V_{IH} = 5V$			1.0			1.0		μs
Write and XFER Pulse Width	t_W	$V_{IL} = 0V$, $V_{IH} = 5V$, $T_A = 25°C$	8	320	60		320	250		ns
		$T_{MIN} \leq T_A \leq T_{MAX}$	10	320	100		500	350		ns
Data Set Up Time	t_{DS}	$V_{IL} = 0V$, $V_{IH} = 5V$, $T_A = 25°C$	10	320	60		320	250		ns
		$T_{MIN} \leq T_A \leq T_{MAX}$		320	100		500	350		ns
Data Hold Time	t_{DH}	$V_{IL} = 0V$, $V_{IH} = 5V$ $T_A = 25°C$	10	90	50		300	200		ns
		$T_{MIN} \leq T_A \leq T_{MAX}$		90	60		350	260		ns
Control Set Up Time	t_{CS}	$V_{IL} = 0V$, $V_{IL} = 5V$, $T_A = 25°C$	10	320	60		320	250		ns
		$T_{MIN} \leq T_A \leq T_{MAX}$		320	100		500	350		ns
Control Hold Time	t_{CH}	$V_{IL} = 0V$, $V_{IH} = 5V$, $T_A = 25°C$	10	10			10			ns
		$T_{MIN} \leq T_A \leq T_{MAX}$		10			10			ns

Note 1: *"Absolute Maximum Ratings"* are those values beyond which the safety of the device cannot be guaranteed. These specifications are not meant to imply that the devices should be operated at these *"Absolute Maximum"* limits.

Note 2: All voltages are measured with respect to GND, unless otherwise specified.

Note 3: This 500 mW specification applies for all packages. The low intrinsic power dissipation of this part (and the fact that there is no way to significantly modify the power dissipation) removes concern for heat sinking.

Note 4: For current switching applications, both I_{OUT1} and I_{OUT2} must go to ground or the "Virtual Ground" of an operational amplifier. The linearity error is degraded by approximately $V_{OS} \div V_{REF}$. For example, if $V_{REF} = 10V$ then a 1 mV offset, V_{OS}, on I_{OUT1} or I_{OUT2} will introduce an additional 0.01% linearity error.

Note 5: Guaranteed at $V_{REF} = \pm 10\,V_{DC}$ and $V_{REF} = \pm 1\,V_{DC}$.

Note 6: $T_{MIN} = 0°C$ and $T_{MAX} = 70°C$ for "LCN" suffix parts.
$T_{MIN} = -40°C$ and $T_{MAX} = 85°C$ for "LCD" suffix parts.
$T_{MIN} = -55°C$ and $T_{MAX} = 125°C$ for "LD" suffix parts.

Note 7: The unit "FSR" stands for "Full Scale Range." "Linearity Error" and "Power Supply Rejection" specs are based on this unit to eliminate dependence on a particular V_{REF} value and to indicate the true performance of the part. The "Linearity Error" specification of the DAC0830 is"0.05% of FSR (MAX)." This guarantees that after performing a zero and full scale adjustment (See Sections 2.5 and 2.6),the plot of the 256 analog voltage outputs will each be within $0.05\% \times V_{REF}$ of a straight line which passes through zero and full scale.

Note 8: This specification implies that all parts are guaranteed to operate with a write pulse or transfer pulse width (t_W) of 320 ns. A typical part will operate with t_W of only 100 ns. The entire write pulse must occur within the valid data interval for the specified t_W, t_{DS}, t_{DH}, and t_S to apply.

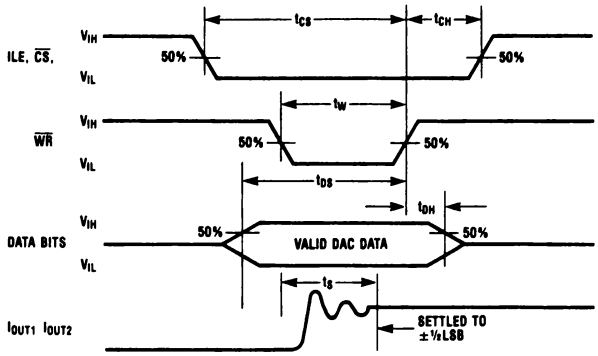
Note 9: To achieve this low feedthrough in the D package, the user must ground the metal lid. If the lid is left floating, the feedthrough is typically 6 mV.

Note 10: Guaranteed by design but not tested.

Note 11: A 100 nA leakage current with $R_{fb} = 20k$ and $V_{REF} = 10V$ corresponds to a zero error of $(100 \times 10^{-9} \times 20 \times 10^3) \times 100/10$ which is 0.02% of FS.

FIG. 44. - Caratteristiche elettriche del DAC 0830.

Switching Waveforms:



Definition of Package Pinouts

Control Signals (All control signals level actuated)

$\overline{CS}$: Chip Select (active low). The $\overline{CS}$ in combination with ILE will enable $\overline{WR_1}$.

ILE: Input Latch Enable (active high). The ILE in combination with $\overline{CS}$ enables $\overline{WR_1}$.

$\overline{WR_1}$: Write 1. The active low $\overline{WR_1}$ is used to load the digital input data bits (DI) into the input latch. The data in the input latch is latched when $\overline{WR_1}$ is high. To update the input latch — $\overline{CS}$ and $\overline{WR_1}$ must be low while ILE is high.

$\overline{WR_2}$: Write 2 (active low). This signal, in combination with $\overline{XFER}$, causes the 8-bit data which is available in the input latch to transfer to the DAC register.

$\overline{XFER}$: Transfer control signal (active low).The $\overline{XFER}$ will enable $\overline{WR_2}$.

Other Pin Functions

DI_0-DI_7: Digital Inputs. DI_0 is the least significant bit (LSB) and DI_7 is the most significant bit (MSB).

I_{OUT1}: DAC Current Output 1. I_{OUT1} is a maximum for a digital code of all 1's in the DAC register, and is zero for all 0's in DAC register.

I_{OUT2}: DAC Current Output 2. I_{OUT2} is a constant minus I_{OUT1}, or $I_{OUT1} + I_{OUT2}$ = constant (I full scale for a fixed reference voltage).

R_{fb}: Feedback Resistor. The feedback resistor is provided on the IC chip for use as the shunt feedback resistor for the external op amp which is used to provide an output voltage for the DAC. This on-chip resistor should always be used (not an external resistor) since it matches the resistors which are used in the on-chip R-2R ladder and tracks these resistors over temperature.

V_{REF}: Reference Voltage Input. This input connects an external precision voltage source to the internal R-2R ladder. V_{REF} can be selected over the range of +10 to −10V. This is also the analog voltage input for a 4-quadrant multiplying DAC application.

V_{CC}: Digital Supply Voltage. This is the power supply pin for the part. V_{CC} can be from +5 to $+15V_{DC}$. Operation is optimum for $+15V_{DC}$.

AGND: Analog Ground. This is the ground for the analog circuitry. This pin must always be connected to the digital ground potential.

DGND: Digital Ground. This is the ground for the digital logic.

FIG. 45. - Forme d'onda e descrizione dei pin d'uscita del DAC 0830.

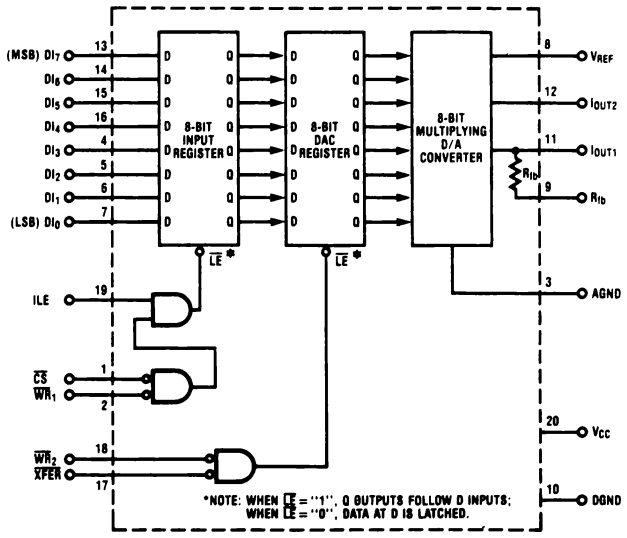
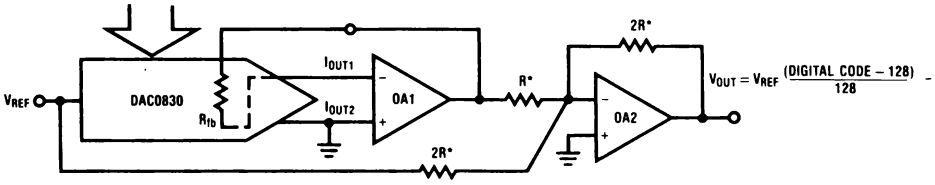


FIG. 46. - Schema funzionale del DAC 0830



*THESE RESISTORS ARE AVAILABLE FROM BECKMAN INSTRUMENTS, INC. AS THEIR PART NO. 694-3-R10K-D

INPUT CODE	IDEAL VOUT	
MSB . . . LSB	+VREF	−VREF
11111111	VREF − 1 LSB	−|VREF| + 1 LSB
10000000	VREF/2	−|VREF|/2
00000000	0	0
01111111	−1 LSB	+1 LSB
00111111	−|VREF|/2 − 1 LSB	|VREF|/2 + 1 LSB
00000000	−|VREF|	+|VREF|

FIG. 47. - DAC 0830 capace di generare una uscita bipolare partendo da una tensione di riferimento fissata.

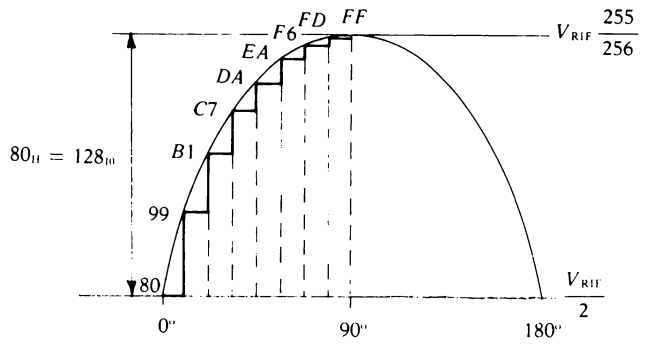


FIG. 48. -

Una approssimazione maggiore alla forma d'onda desiderata può essere raggiunta suddividendo in un numero maggiore di passi la tabulazione dei valori del quarto di periodo (fino ad un massimo di 128).

Una forma d'onda sinusoidale a valore medio uguale a zero può invece ottenersi mediante l'impiego di un ulteriore amplificatore operazionale come mostrato in fig. 47 oppure più semplicemente con un condensatore di by-pass collegato all'uscita del μA 741.

Il dente di sega viene invece generato ogni qualvolta un impulso negativo viene inviato al pin $\overline{\text{INT}}$ della CPU programmata nel modo 1 di interruzione. Il programma relativo alla generazione del dente di sega, sotto forma di subroutine d'interrupt, è molto semplice limitandosi ad un incremento del contenuto dell'accumulatore seguito da una istruzione di OUT.

Software:

TAB HL (0400): 80,99, B1, C7, DA, EA, F6, FD, FF.

0100	31 00 08	LD SP, 0800	Definisci l'area di stack
0103	3E C3	LD A, C3	Carica l'indirizzo della
0105	32 38 00	LD (0038), A	subroutine d'interruzione
0108	FD 21 00 02	LD IY, 0200	per il modo 1
010C	FD 22 39 00	LD (0039), IY	
0110	06 09	LD B, 09	Carica il contatore di valori
0112	21 00 04	LD HL, 0400	Carica l'indirizzo della tabella

P_1	0115	7E	LD A, (HL)	Carica in accumulatore il contenuto di HL
	0116	C3 19 01	JP P_2	Ritarda di 10 cicli per compensare quello introdotto dalle istruzioni CPL ed INC A
P_2	0119	D3 40	OUT (40), A	Invia al DAC il segnale letto
	011B	23	INC HL	Incrementa HL
	011C	05	DEC B	Decrementa il contatore
	011D	C2 15 01	JP NZ, P_1	Salta a P_1 se non sono stati letti tutti i valori della tabella
	0120	06 08	LD B, 08	Carica il contatore di valori
	0122	21 07 04	LD HL, 0407	Punta alla locazione di memoria 0407
P_3	0125	7E	LD A, (HL)	Leggi il valore
	0126	C3 29 01	JP P_4	
P_4	0129	D3 40	OUT (40), A	
	012B	2B	DEC HL	
	012C	05	DEC B	
	012D	C2 25 01	JP NZ, P_3	Salta a P_3 se B è diverso da zero.
	0130	06 08	LD B, 08	Carica il contatore di valori
	0132	21 01 04	LD HL, 0401	Punta alla locazione 0401

P_5	0135	7E	LD A, (HL)	Leggi il valore
	0136	2F	CPL	Complementa a due il valore letto
	0137	3C	INC A	
	0138	D3 40	OUT (40), A	Invia al DAC il valore letto
	013A	23	INC HL	
	013B	05	DEC B	
	013C	C2 35 01	JP NZ, P_5	
	013F	06 08	LD B,08	
	0141	21 07 04	LD HL, 0407	
P_6	0144	7E	LD A, (HL)	
	0145	2F	CPL	
	0146	3C	INC A	
	0147	D3 40	OUT (40), A	
	0149	2B	DEC HL	
	014A	05	DEC B	
	014B	C2 44 01	JP NZ, P_6	
	014E	06 08	LD B, 08	
	0150	21 01 04	LD HL, 0401	
	0153	FB	EI	Abilita le interruzioni
	0154	ED 56	IM 1	Programma le interruzioni nel modo 1
	0156	C3 15 01	JP P_1	Inizia un altro ciclo

μA741

FREQUENCY-COMPENSATED OPERATIONAL AMPLIFIER

FAIRCHILD LINEAR INTEGRATED CIRCUIT

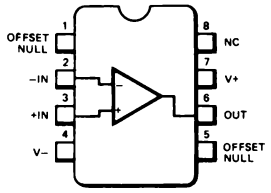
GENERAL DESCRIPTION – The μA741 is a high performance monolithic Operational Amplifier constructed using the Fairchild Planar* epitaxial process. It is intended for a wide range of analog applications. High common mode voltage range and absence of latch-up tendencies make the μA741 ideal for use as a voltage follower. The high gain and wide range of operating voltage provides superior performance in integrator, summing amplifier, and general feedback applications. Electrical characteristics of the μA741A and E are identical to MIL-M-38510/10101.

- **NO FREQUENCY COMPENSATION REQUIRED**
- **SHORT CIRCUIT PROTECTION**
- **OFFSET VOLTAGE NULL CAPABILITY**
- **LARGE COMMON MODE AND DIFFERENTIAL VOLTAGE RANGES**
- **LOW POWER CONSUMPTION**
- **NO LATCH-UP**

ABSOLUTE MAXIMUM RATINGS

Supply Voltage	
μA741A, μA741, μA741E	±22 V
μA741C	±18 V
Internal Power Dissipation (Note 1)	
Metal Can	500 mW
Molded and Hermetic DIP	670 mW
Mini DIP	310 mW
Flatpak	570 mW
Differential Input Voltage	±30 V
Input Voltage (Note 2)	±15 V
Storage Temperature Range	
Metal Can, Hermetic DIP, and Flatpak	−65°C to +150°C
Mini DIP, Molded DIP	−55°C to +125°C
Operating Temperature Range	
Military (μA741A, μA741)	−55°C to +125°C
Commercial (μA741E, μA741C)	0°C to +70°C
Lead Temperature (Soldering)	
Metal Can, Hermetic DIPs, and Flatpak (60 s)	300°C
Molded DIPs (10 s)	260°C
Output Short Circuit Duration (Note 3)	Indefinite

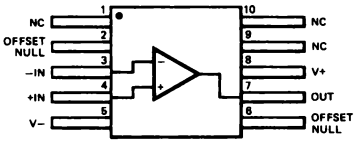
8-LEAD MINIDIP
(TOP VIEW)
PACKAGE OUTLINES 6T 9T
PACKAGE CODES T R



ORDER INFORMATION

TYPE	PART NO.
μA741C	μA741TC
μA741C	μA741RC

10-LEAD FLATPAK
(TOP VIEW)
PACKAGE OUTLINE 3F

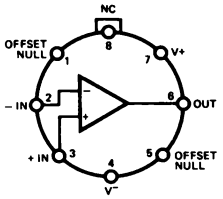


ORDER INFORMATION

TYPE	PART NO.
μA741A	μA741AFM
μA741	μA741FM

CONNECTION DIAGRAMS

8-LEAD METAL CAN
(TOP VIEW)
PACKAGE OUTLINE 5B

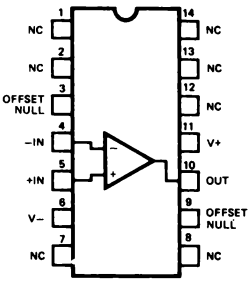


Note: Pin 4 connected to case

ORDER INFORMATION

TYPE	PART NO.
μA741A	μA741AHM
μA741	μA741HM
μA741E	μA741EHC
μA741C	μA741HC

14-LEAD DIP
(TOP VIEW)
PACKAGE OUTLINE 6A, 9A



ORDER INFORMATION

TYPE	PART NO.
μA741A	μA741ADM
μA741	μA741DM
μA741E	μA741EDC
μA741C	μA741DC
μA741C	μA741PC

FIG. 49. - Piedinatura e generalità dell'amplificatore operazionale μA 741.

FAIRCHILD LINEAR INTEGRATED CIRCUITS • μA741

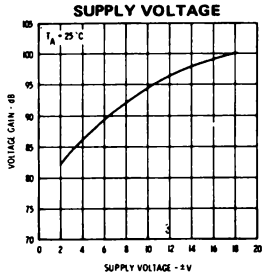
μA741C

ELECTRICAL CHARACTERISTICS (V_S = ±15 V, T_A = 25°C unless otherwise specified)

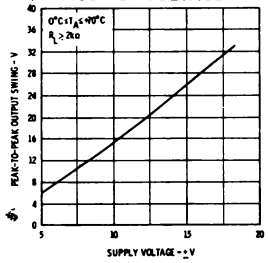
PARAMETERS (see definitions)		CONDITIONS	MIN	TYP	MAX	UNITS
Input Offset Voltage		$R_S \leq 10$ kΩ		2.0	6.0	mV
Input Offset Current				20	200	nA
Input Bias Current				80	500	nA
Input Resistance			0.3	2.0		MΩ
Input Capacitance				1.4		pF
Offset Voltage Adjustment Range				±15		mV
Input Voltage Range			±12	±13		V
Common Mode Rejection Ratio		$R_S \leq 10$ kΩ	70	90		dB
Supply Voltage Rejection Ratio		$R_S \leq 10$ kΩ		30	150	μV/V
Large Signal Voltage Gain		$R_L \geq 2$ kΩ, V_{OUT} = ±10 V	20,000	200,000		
Output Voltage Swing		$R_L \geq 10$ kΩ	±12	±14		V
		$R_L \geq 2$ kΩ	±10	±13		V
Output Resistance				75		Ω
Output Short Circuit Current				25		mA
Supply Current				1.7	2.8	mA
Power Consumption				50	85	mW
Transient Response (Unity Gain)	Rise time	V_{IN} = 20 mV, R_L = 2 kΩ, $C_L \leq$ 100 pF		0.3		μs
	Overshoot			5.0		%
Slew Rate		$R_L \geq 2$ kΩ		0.5		V/μs
The following specifications apply for 0°C ≤ T_A ≤ +70°C:						
Input Offset Voltage					7.5	mV
Input Offset Current					300	nA
Input Bias Current					800	nA
Large Signal Voltage Gain		$R_L \geq 2$ kΩ, V_{OUT} = ±10 V	15,000			
Output Voltage Swing		$R_L \geq 2$ kΩ	±10	±13		V

TYPICAL PERFORMANCE CURVES FOR μA741E AND μA741C

OPEN LOOP VOLTAGE GAIN AS A FUNCTION OF SUPPLY VOLTAGE



OUTPUT VOLTAGE SWING AS A FUNCTION OF SUPPLY VOLTAGE



INPUT COMMON MODE VOLTAGE RANGE AS A FUNCTION OF SUPPLY VOLTAGE

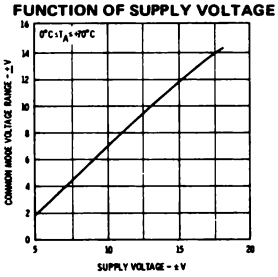


FIG. 50. - Caratteristiche elettriche del μA 741.

ESERCIZI PROPOSTI

1. Progettare una rete comparativa in grado di riconoscere una parola proveniente da un bus a quattro linee utilizzando switches e porte logiche Xor ed Or.

2. Determinare il software ed hardware necessari per scrivere nelle sedici locazioni di memoria della SN 7489 (Vol. Elettronica Digitale cap. 6°) i primi sedici numeri esadecimali forniti dal bus degli indirizzi dello Z 80.

3. Scrivere il software capace di verificare il corretto funzionamento dei motori passo-passo dello schema di fig. 28.

4. Progettare un circuito in grado di generare uno stato di attesa ad ogni ciclo di memoria.

5. Utilizzando un PIO Z 80 ed un 555 determinare il software e l'hardware necessari per generare un tempo di ritardo variabile.

6. È vantaggioso l'utilizzo della istruzione OTIR nel main-program di pag. 319?

CAPITOLO SETTIMO

PROGETTO DI UN MICROCOMPUTER PER USI GENERALI

1. Introduzione

Lo scopo del presente capitolo è di chiarire le problematiche concernenti la effettiva connessione dei vari *chips* necessari alla realizzazione di un semplice sistema a microprocessore. Molto spesso infatti, nonostante che i costruttori ne descrivano tutte le caratteristiche, certi fatti, relativi alla loro interconnessione, sono non del tutto chiari. Un modo per scavalcare ostacoli di questo tipo potrebbe essere l'utilizzazione di schede SBC (Single Board Computer) che però obbliga ad una scelta più o meno rigida del particolare μP e delle strutture offerte dal costruttore, oltre che ad un costo elevato.

Si ritiene utile perciò, l'esecuzione del progetto di un semplice microcomputer per scopi generali.

Come già ampliamente chiarito in precedenza, la sua costituzione richiede la presenza delle seguenti unità:
1) un generatore di clock;
2) una unità centrale di processo;
3) una memoria RAM;
4) una memoria ROM;
5) un dispositivo di ingresso/uscita che consenta di dialogare con il sistema.

Nella pratica, alcune di queste unità possono essere omesse. Ad esempio, in un controllo di processo può risultare mancante sia la memoria RAM che la tastiera, poiché il programma è interamente inserito nella ROM, e non occorre un'interfaccia con l'operatore (tipicamente costituita da tastiera e display).

Il sistema che si vuole realizzare è modulare, per facilitare successive espansioni, ed è schematicamente rappresentato in figura 1.

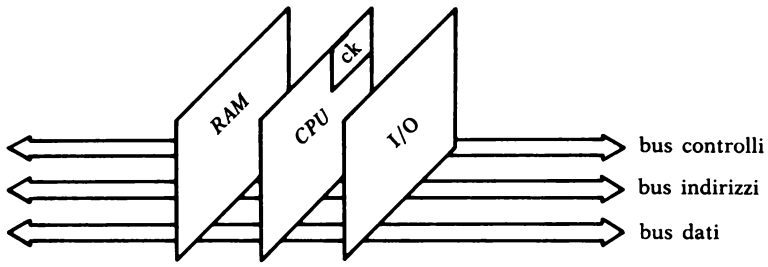


FIG. 1. - Montaggio modulare per il microcomputer.

Come si osserva è mancante la scheda contenente la memoria ROM, poiché, come si vedrà, non è inizialmente necessaria. Infatti uno degli scopi del progetto è quello di realizzare passo-passo la scrittura e la verifica di un software (monitor) che, successivamente caricato in ROM, permetta la supervisione del sistema consentendo l'esecuzione di un qualsiasi programma senza utilizzare supporti esterni. Ciò significa che si procederà costruendo dapprima l'hardware sulla base delle specifiche esigenze ed eseguendo, successivamente, una serie di operazioni che potrebbero essere le seguenti:
all'atto dell'accensione la CPU, tramite un comando esterno, viene tenuta sconnessa dai vari bus e mediante una tastiera formata da interruttori (poiché non c'è ancora un programma che la legga e la decodifichi), viene memorizzato nella RAM un semplice programma di diagnostica, quale ad esempio il caricamento di un dato in una locazione di memoria. Scopo di questo primo programma di poche istruzioni è quello di verificare la correttezza dell'hardware.

Il caricamento può avvenire, ad esempio, impostando sulla consolle con sedici interruttori l'indirizzo, con otto interruttori il dato o l'istruzione e successivamente dando un comando di scrittura (write) alla memoria RAM tramite un altro pulsante.

Ripetendo in modo identico le precedenti operazioni si procede alla completa scrittura del programma. È bene prevedere inoltre sulla consolle, la presenza di una serie di display per la visualizzazione dei dati ed indirizzi impostati oltre che dei risultati delle elaborazioni.

Il modo più semplice per dare inizio alla esecuzione del programma è di riconnettere la CPU ai vari bus e di caricare il suo *program counter* con l'indirizzo della prima locazione di memoria ove è immagazzinato il programma. Ciò può effettuarsi in vari modi secondo il microprocessore utilizzato (tramite il RESET o anche tramite l'interruzione), ma quello più semplice utilizza la facoltà, comune a quasi tutti, di azzerare il PC allorché viene dato il comando di RESET. Questo significa che, non appena ricevuto tale comando, la CPU legge il contenuto della locazione zero della

memoria. Nel caso dello Z 80 in tale locazione deve essere caricato un indirizzo che generalmente corrisponde all'indirizzo di inizio del *monitor*, ma che nel caso attuale è la locazione di partenza del precedente programma di test.

Naturalmente i programmi per saggiare l'esattezza di tutto l'hardware possono essere più o meno numerosi e complessi fino a quando si è raggiunta la sicurezza di un corretto montaggio.

Una volta terminata la fase di prova, si può iniziare la scrittura vera e propria dei vari pacchetti software che costituiscono il sistema operativo.

2. La scelta del microprocessore

I criteri dei quali tener conto nella scelta del microprocessore sono numerosi e molto diversificati tra di loro poiché dipendono dal tipo di impiego che dovrà svolgere.

Una loro prima elencazione può risultare la seguente:
1) il costo, che riveste particolare importanza per elevati livelli di produzione ed è direttamente legato alla tecnologica, velocità e potenza di calcolo.
2) la velocità. Una misura piuttosto precisa della velocità di un processore può essere effettuata mediante l'esecuzione di programmi di test (programmi di Benchmark) di almeno un centinaio di istruzioni, possibilmente attinenti al reale problema da trattare.

L'esecuzione del programma di test può risultare utile anche per giudicare eventuali difficoltà nella lettura dei manuali ed impiego di istruzioni. Programmi di Benchmark sono ad esempio la conversione di 512 bytes dal codice ASCII al codice EBCDIC, oppure lo spostamento di 256 bytes dal registro A al registro B e via di seguito.
3) la potenza di calcolo, che è legata al numero di bit trattati in parallelo ed al set di istruzioni.
4) la documentazione, supporto necessario per la stesura delle specifiche.
5) i circuiti di supporto. Questi sono diventati quasi altrettanto potenti che le CPU stesse e le mettono in grado di eseguire meglio la manipolazione dei dati per l'analisi ed il processo.

I circuiti di supporto possono dividersi in tre categorie:
a) i circuiti periferici, che eseguono comandi o forniscono dati al sistema e vengono trattati come memory-mapped o porti di I/0;
b) processori schiavi, che eseguono particolari subroutine;
c) coprocessori, che lavorano in parallelo con la CPU principale.

Nella seguente tabella sono riportati i μP attualmente sul mercato con alcune delle loro caratteristiche più importanti.

Nel progetto attuale, restando fedeli all'impostazione del testo, si è scelto lo Z 80 che è già stato ampliamente utilizzato.

General-purpose microprocessors

Word size (data/instruction)	Original source manufacturer	Processor	Process technology	Direct addressing range (words)	Number of basic instructions	Maximum clock frequency (MHz)/phases	Instruction time shortest/longest[2] (μs)	TTL compatible	BCD arithmetic	On-chip interrupts/levels	Number of internal general-purpose registers	Number of stack registers
1/4	Motorola	MC14500	CMOS	0	16	1/1	1/1	Yes	No	Yes/1	1	0
4/8	Fujitsu	MB8849	NMOS	2048	70	2	3/6	Yes	Yes	Yes/2	128	4
4/8	Intel	4004	PMOS	4 k	46	0.74/2	10.8/21.6	No	Yes	Yes/1	16	3×12
4/8	Intel	4040	PMOS	8 k	60	0.74/2	10.8/21.6	No	Yes	Yes/1	24	7×12
4/8	National Semiconductor	COP402	NMOS	1 k	49	1/1	4	Yes	Yes	Yes/3	64×4	RAM
4/8	National Semiconductor	COP402M	NMOS	1 k	49	1/1	4	Yes	Yes	No	64×4	RAM
4/8	National Semiconductor	COP404L	NMOS	2 k	49	0.25/1	16	Yes	Yes	Yes/3	128×4	RAM
4/8	NEC Microcomputers	μPD556	PMOS	2 k	80	0.44/1	4.5/9	Yes	Yes	Yes/2	96×4	3
4/8	OKI	MSM5840E	CMOS	4 k	98	2/1	16/32	Yes	Yes	Yes/2	128×4	4
4/8	Panasonic	MN1498	NMOS	1 k	66	0.3/1	10/20	Yes	Yes	Yes/1	64×4	RAM
4/8	Panasonic	MN1499	NMOS	2 k	75	0.3/1	10/20	Yes	Yes	Yes/1	64×4	RAM
4/8	Panasonic	MN1499A	NMOS	2 k	75	0.3/1	10/20	Yes	Yes	Yes/1	128×4	RAM
4/8	Panasonic	MN1599	NMOS	4 k	125	1/1	2/4	Yes	Yes	Yes/4	256×4	RAM
4/8	Texas Instruments	TMS1099SE	PMOS	1 k	43	0.40/1	15/15	Yes	Yes	No	66×4	1×10
4/8	Texas Instruments	TMS1098SE	PMOS	2 k	40	0.40/1	15/15	Yes	Yes	No	130×4	1×11
4/8	Texas Instruments	TMS1097SE	PMOS	4 k	41	0.55/1	11/11	Yes	Yes	No	130×4	3×12
4/8	Texas Instruments	TMS1096JL	PMOS	2 k	45	0.55/1	11/11	Yes	Yes	Yes/1	130×4	4×11
4/8	Toshiba	TMP4300	NMOS	2 k	67	1/1	33/360	Yes	Yes	Yes/1	128×4	8
4/8	Toshiba	TCP4600	CMOS	4 k	52	0.1/1	20/40	Yes	Yes	Yes/2	160×4	2
4/10	NEC Microcomputers	μPD555	PMOS	1920×10	72	0.2/1	10/20	Yes	Yes	Yes/2	96×4	4
8/8	Fairchild	2 chip F8 (3850)	NMOS	64 k	69	2/1	2/13	Yes	Yes	Yes/1	64	RAM
8/8	General Instrument	8000	PMOS	1 k	48	0.8/2	1.25/3.75	No	Yes	Yes/1	48	0
8/8	Intel	8008	PMOS	16 k	48	0.8/2	12.5/37.5	No	Yes	Yes/1	6	7×14
8/8	Intel	i8031	NMOS	128 k	111	12	1/4	Yes	Yes	Yes/2	128	RAM
8/8	Intel	8035/8039	NMOS	64 k	96	6/1	2.5/5	Yes	Yes	Yes/1	64	RAM
8/8	Intel	8080A	NMOS	64 k	78	2.6/2	1.5/3.75	Yes[3]	Yes	Yes/1	8	RAM
8/8	Intel	8085	NMOS	64 k	80	5.5/1	0.8/5.2	Yes	Yes	Yes/4	8	RAM
8/8	Intersil	80C35	CMOS	64 k	96	6/1	2.5/5	Yes	Yes	Yes/1	64	RAM
8/8	MOS Technology	MCS-650X	NMOS	64 k	56	4/1	0.5/3.5	Yes	Yes	Yes/1	0	RAM
8/8	MOS Technology	MCS-651X	NMOS	64 k	56	4/2	0.5/3.5	Yes	Yes	Yes/1	0	RAM
8/8	Mostek	MK38P70/02	NMOS	4 k	70+	4/1	1/6.5	Yes	Yes	Yes/4	64	RAM
8/8	Mostek	MK38P73/02	NMOS	4 k	70+	4/1	1/6.5	Yes	Yes	Yes/4	64	RAM
8/8	Motorola	M6800	NMOS	64 k	72	2/2	1/2.5	Yes	Yes	Yes/1	0	RAM
8/8	Motorola	6802/6808	NMOS	64 k	72	2/1	2/5	Yes	Yes	Yes/1	128/0	RAM
8/8	Motorola	6803	NMOS	64 k	82	3.58/1	2/12	Yes	Yes	Yes/1	128	RAM
8/8	Motorola	M6809	NMOS	64 k	59	2/1	2/5	Yes	Yes	Yes/1	0	RAM
8/8	National Semiconductor	INS8060	NMOS	64 k	46	4/1	5/22	Yes	Yes	Yes/1	8	RAM
8/8	National Semiconductor	INS8040	NMOS	64 k	96	11/1	1.4/2.8	Yes	Yes	Yes/1	256	RAM
8/8[7]	National Semiconductor	INS8070	NMOS	64 k	74	4/1	3/1000[8]	Yes	No	Yes/2	9	RAM
8/8	National Semiconductor	INS8073	NMOS	64 k	see comments	4/1	3/1000	Yes	Yes	Yes/2	64	RAM
8/8	National Semiconductor	NSC800	CMOS	64 k	150+	8/1	0.5/2.88	Yes	Yes	Yes/5	14	RAM
8/8	NEC Microcomputers	μPD7800	NMOS	64 k	140	1/1	2/4	Yes	Yes	Yes[5]	128+16	RAM

Tabella 1

On-chip clock	DMA capability	Specialized memory & I/O circuits avail.	Prototyping system avail.	Package size (pins)	Voltages required (V)	Assembly language development system	High-level languages	Time-sharing cross software	Comments
Yes	No	No[4]	No	16	3 to 18	No	No	No	Needs external program counter
Yes	No	No	Yes	64	5	Yes	No	Yes	ROM-less emulator chip for MB8840 family of all-in-one processors
No	No	Yes	No	16	15	Yes	Yes	Yes	Superseded by 4040
No	No	Yes	Yes	24	15	Yes	Yes	Yes	General-purpose 4-bit μP
Yes	No	Yes	Yes	40	4.5 to 6.3	Yes	No	Yes	ROM-less version of COP420 and 440
Yes	No	Yes	Yes	40	4.5 to 6.3	Yes	No	Yes	single chip μC w/serial I/O, 20 I/O lines, event-counting
Yes	No	Yes	Yes	40	4.5 to 9.5	Yes	No	Yes	
Yes	No	No	Yes	64	−10	Yes	No	Yes	ROM-less version of μPD546
Yes	No	Yes	Yes	42	5	Yes	No	Yes	ROM-less evaluation chip; includes 30 I/O lines, 8-bit timer/counter, and draws 0.8 mA
Yes	No	Yes	Yes	40	5	Yes	No	Yes	ROM-less version of MN1402, but 66 instructions and comes in a 40-pin package
Yes	No	Yes	Yes	64	5	Yes	No	Yes	ROM-less version of MN1400
Yes	No	Yes	Yes	64	5	Yes	No	Yes	ROM-less version of MN1405, but 128 nibbles of on-chip RAM
Yes	Yes	Yes	Yes	64	5	Yes	No	Yes	ROM-less version of MN1564; chip has 12 4-bit I/O ports
Yes	No	Yes	Yes	64	−9	Yes	No	Yes	ROM-less version of TMS1000/1200 microcomputer
Yes	No	Yes	Yes	64	−9	Yes	No	Yes	ROM-less version of TMS1100/1300 microcomputer
Yes	No	Yes	Yes	64	−9	Yes	No	Yes	ROM-less version of TMS1400 microcomputer
Yes	No	Yes	Yes	64	−9	Yes	No	Yes	ROM-less version of TMS2100/2300 microcomputer
Yes	No	Yes	Yes	64	5	Yes	No	Yes	ROM-less evaluator chip; uses a 24-bit control word
Yes	No	Yes	Yes	64	5	Yes	No	Yes	ROM-less evaluator chip for TCP 4600 microcomputer family
Yes	No	No	Yes	64	−10	Yes	No	Yes	ROM-less version of μPD548
Yes	No	Yes	Yes	40	5,12	Yes	Yes	Yes	Usually used with program storage unit
No	No	Yes	Yes	40	5,−12	No	Yes	Yes	Predecessor of F8
No	No	Yes	Yes	18	5,−9	Yes	Yes	Yes	Predecessor of 8080, still in wide use
Yes	No	Yes	Yes	40	5	Yes	No	Yes	ROM-less version of 8051 all-in-one processor
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less versions of 8048/8049
No	Yes	Yes	Yes	40	5,12,−5	Yes	Yes	Yes	By and large, still the most popular
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	8080 code compatible, has built-in clock
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less version of 80C48 all-in-one processor
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	Provides 13 addressing modes
No	No	Yes	Yes	40	5	Yes	Yes	Yes	Similar to 650X but needs 2φ clock
Yes	No	Yes	Yes	40+	5	Yes	Yes	Yes	ROM-less piggyback package version of MK3870/12 all-in-one processor
Yes	No	Yes	Yes	40+	5	Yes	Yes	Yes	ROM-less piggyback package version of MK3873/12 all-in-one processor
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Available in depletion-load version
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	6802 has 128x8 on chip RAM; 6808 has no RAM
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less version of 6801 single-chip μC
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Enhanced 6800 command set
Yes	Yes	No[4]	No	40	5	Yes	Yes	Yes	Has handy daisy-chain capability
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less version of INS8050 single chip μC
Yes	Yes	Yes[4]	Yes	40	5	Yes	Yes	No	ROM-less version of INS8072; 64 bytes of on chip RAM
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	On-chip ROM contains Tiny Basic interpreter (nibble 2); 74 assembly-level commands also available
Yes	Yes	Yes	Yes	40	3 to 12	Yes	Yes	Yes	Executes Z80 instructions and has 8085 bus structure
Yes	Yes	Yes	Yes	64Q	5	Yes	No	Yes	ROM-less version of μPD7801 microcomputer

General-purpose microprocessors (Continued)

Word size (data/instruction)	Original source manufacturer	Processor	Process technology	Direct addressing range (words)	Number of basic instructions	Maximum clock frequency (MHz)/phases	Instruction time shortest/longest[2] (μs)	TTL compatible	BCD arithmetic	On-chip interrupts/levels	Number of internal general-purpose registers	Number of stack registers
8/8	RCA	1802	CMOS	64 k	91	5/1	3.2/4.8	Yes	No	Yes/1	16	RAM
8/8	RCA	CPD1805	CMOS/SOS	64 k	113	8000/1	2/3	Yes	No	Yes/1	16×8	RAM
8/8	Signetics	2650A	NMOS	32 k	75	2/1	1.5/6	Yes	Yes	Yes/1	7	8×15
8/8	Zilog	Z80B	NMOS	64 k	150+	6/1	0.7/4.2	Yes	Yes	Yes/1	14	RAM
8/8	Zilog	Z8602	NMOS	126 k	47	8/1	1.5/3.75	Yes	Yes	Yes/6	144×8	RAM
8/8	Zilog	Z8603	NMOS	126 k	47	8/1	1.5/3.75	Yes	Yes	Yes/6	144×8	RAM
8/8	Zilog	Z8681	NMOS	126 k	47	8/1	1.5/3.75	Yes	Yes	Yes/6	144×8	RAM
8/16	Signetics	8X300	Bipolar	8 k	NA[10]	5/1	0.2	Yes	No	No	8	0
12/12	Intersil	6100	CMOS	4 k	81	3.3/1[12]	2.5/5.5	Yes	No	Yes/1	0	RAM
12/12	Toshiba	T3535	PMOS NMOS	4 k	108	2.5/1	10/30	Yes	No	Yes/8	8	RAM
16/16	Advanced Micro Devices	Am29116	ECL	64 k	30+	10/1	0.1/0.2	Yes	No	No	32	32
16/16	Data General	mN601	NMOS	32 k	42	8.33/2	1.2/29.5	Yes	No	Yes/1	4	RAM
16/16	Data General	mN602	NMOS	64 k	82	8.3/2	2.4/53	Yes	No	Yes/16	4[6]	RAM
16/16	Fairchild	9445	I^3L	64 k	100	20/1[9]	0.3/5.7	Yes	No	Yes/16	4	RAM
16/16	Ferranti	F100L	Bipolar	32 k	153	14/1	1.19/14	Yes	No	Yes/1	RAM	RAM
16/16	General Instrument	CP1600/1610	NMOS	64 k	87	4/2	1.6/4.8	Yes	No	Yes/1	8	RAM
16/16	Intel	iAPX86/10	NMOS	IM[5]	97	5/1	0.4/37.8	Yes	Yes	Yes/1	8	RAM
16/16[1]	Intel	iAPX88/10	NMOS	64 k[5]	97	5/1	0.4/37.8	Yes	Yes	Yes/1	8	RAM
16/16	Motorola	MC68000	NMOS	16M[5]	61	8/1	0.5/NA[10]	Yes	Yes	Yes/1	16	RAM
16/16	National Semiconductor	INS8900	NMOS	64 k	45	2/1	2.5/5	Yes	Yes	Yes/6	4	10×16
16/16[1]	National Semiconductor	NS16008	NMOS	64 k[5]	78/100+	NA[10]	NA[10]	Yes	Yes	Yes	8	RAM
16/16	National Semiconductor	NS16016	NMOS	64 k	78/100+	NA[10]	NA[10]	Yes	Yes	Yes	8	RAM
16/16	National Semiconductor	NS16032	NMOS	16M[5]	100+	NA[10]	NA[10]	Yes	Yes	Yes	8	RAM
16/16	Panafacom	MN1610	NMOS	64 k	33	2/2	2/6	Yes[3]	No	Yes/3	5	RAM
16/16[1]	Texas Instruments	TMS9980/9981	NMOS	8 k	69	4/4	3.2/49.6	Yes[3]	No	Yes/4	16	RAM
16/16[1]	Texas Instruments	TMS9995	NMOS	32 k	72	6/1	1/40	Yes	No	Yes/7	256×8	RAM
16/16	Texas Instruments	TMS/SBP9900	NMOS/I^2L	32 k	69	4/4	2/31	Yes[3]	No	Yes/16	16	RAM
16/16	Western Digital	WD-16	NMOS	64 k	116	3.3/4	2.1/780	Yes	Yes	Yes/16	6	RAM
16/16	Western Digital	Pascal Microengine	NMOS	64 k	150+	3/4	2.4/300[8]	Yes	Yes	Yes/4	RAM	RAM
16/16	Zilog	Z8001	NMOS	48M[5]	110+	8/1	0.75/90	Yes	Yes	Yes/1	16	RAM
16/16	Zilog	Z8002	NMOS	354 k	110+	8/1	0.75/90	Yes	Yes	Yes/1	16	RAM
80/16	Intel	i8087	NMOS	1 Mbyte	46	5	9/100	Yes	No	Yes/1	8×80	8×80

1. Has 8-bit external buses and 16-bit internal buses. 2. With maximum clock. 3. Except clock lines.
4. Standard TTL or MOS circuits will suffice. 5. Range in bytes. 6. Frame Pointer too.
7. Double-precision 16-bit operations available. 8. String search.
9. Clock internally divided by 4 or 6 depending on instruction. 10. Not applicable. 11. 9980 only.
12. At 5 V; as supply voltage increases, clock freq. can increase.

On-chip clock	DMA capability	Specialized memory & I/O circuits avail.	Prototyping system avail.	Package size (pins)	Voltages required (V)	Assembly language development system	High-level languages	Time-sharing cross software	Comments
Yes	Yes	Yes	Yes	40	4 to 10.5	Yes	Yes	Yes	On-chip DMA counter
Yes	Yes	Yes	Yes	40	4 to 10.5	Yes	Yes	Yes	ROM-less version of CDP1804; has 64 bytes of RAM on chip and an 8-bit counter/timer
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	There are 1.25 and 2 MHz versions
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	8080 instructions are a subset
Yes	No	Yes	Yes	64Q	5	Yes	Yes	Yes	ROM-less version of Z8601 microcomputer; all ROM address and data lines available on the 64 pins of the QUIL package
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	Piggyback ROM-less package, otherwise same as 8602
Yes	No	Yes	Yes	40	5	Yes	Yes	Yes	ROM-less and RAM-less version of Z8 microcomputer; still has two counter/timers, serial I/O port, 32 I/O lines
Yes	No	Yes	Yes	50	5	Yes	No	Yes	Intended for high speed controllers
Yes	Yes	Yes	Yes	40	4 to 11	Yes	Yes	Yes	Emulates PDP-8 instruction set
Yes	Yes	Yes	Yes	36	5,−5	Yes	Yes	Yes	Has multiply and divide inst.
No	Yes	Yes	Yes	48	5	Yes	No	Yes	Control-oriented microprogrammable CPU, can generate CRC bits
Yes	Yes	Yes	No	40	5,10,14,−4.25	Yes	Yes	Yes	Emulates NOVA instruction set
No	Yes	Yes	Yes	40	3,12,±5	Yes	Yes	Yes	Executes the NOVA instruction set and addresses double the memory of the mN601
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Executes NOVA 3 and 4 instruction sets; in devel.
No	Yes	Yes	Yes	40	5,1.2	Yes	Yes	Yes	Can do double word operations
No	Yes	Yes	Yes	40	5,12,−3	Yes	Yes	Yes	All internal registers can be accumulators
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Has 24 addressing modes
Yes	Yes	Yes	Yes	40	5	Yes	Yes	Yes	8-bit bus version of 8086 microprocessor
No	Yes	Yes	Yes	64	5	Yes	Yes	Yes	Has 32-bit wide internal structure
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Architecture intended for data handling
No	Yes	Yes	Yes	40	5	Yes	Yes	No	8-bit bus version of dual language (8080/native) CPU, has internal 16-bit bus
No	Yes	Yes	Yes	40	5	Yes	Yes	No	Full 16-bit version, offers 8080A and native instruction sets
No	Yes	Yes	Yes	48	5	Yes	Yes	No	Expanded 16-bit version with eight 32-bit registers, six 24-bit registers and two 16-bit registers; can address 16 Mbytes
No	Yes	Yes	No	40	5,12,−3	Yes	No	No	
Yes	Yes	Yes	No	40	5,12,−5[11]	Yes	Yes	Yes	The 9980 requires external clock
Yes	Yes	Yes	No	40	5	Yes	Yes	Yes	CPU has 256 bytes of RAM on-board to speed context switches
No	Yes	Yes	No	64	5,12,−5	Yes	Yes	Yes	Emulates 990 mini instructions
No	Yes	Yes	Yes	40	5,12,−5	Yes	Yes	No	Very similar to DEC LSI-11
No	Yes	Yes	Yes	40	±12,±5	Yes	Yes	Yes	Five-chip set directly executes Pascal p-code
No	Yes	Yes	Yes	48	5	Yes	Yes	Yes	Address space is divided into segments
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Nonsegmented version
No	Yes	Yes	Yes	40	5	Yes	Yes	Yes	Numeric processor does IEEE floating-point calculations as well as trig, log, and exponential operations

3. Il clock del sistema

In un circuito digitale l'oscillatore è uno dei costituenti più importanti poiché l'accuratezza delle temporizzazioni totali del sistema dipende interamente dalla precisione e dalla costanza della frequenza del clock principale. In questo paragrafo vengono esposte le considerazioni generali per il progetto e la realizzazione pratica di clock a CMOS pilotati al quarzo.

Non tutti i microprocessori richiedono lo stesso circuito di clock. Per alcuni è richiesta solamente l'aggiunta di un cristallo o una rete R-C esterna per stabilire la esatta frequenza di lavoro, mentre per altri può essere necessario un clock a due, tre o quattro fasi anche non simmetriche. Ad esempio nella figura 2 sono mostrate le specifiche richieste dall'8080 della Intel per operare a 2 MHz.

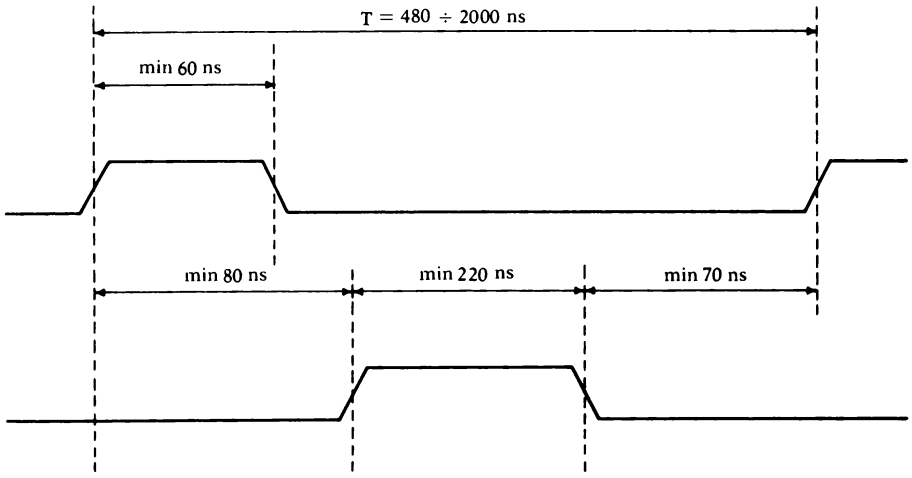


FIG. 2. - Specifiche di clock richieste dall'8080 (Intel).

Particolare importanza rivestono i limiti superiore ed inferiore di frequenza, per cui se il μP opera nelle loro vicinanze, è necessario ridurre la deriva di frequenza del clock principale mediante un controllo a cristallo di quarzo.

Alcuni microprocessori costruiti in tecnologia CMOS, sono statici e possono operare a frequenze bassissime, fino al limite della corrente continua e ciò si rivela molto utile se si desidera seguire il flusso dei dati ad ogni singolo ciclo macchina.

Il circuito fondamentale per un oscillatore consiste di un amplificatore ed una sezione di reazione come è mostrato nella figura 3. Affinché si originino le oscillazioni, è necessario che il prodotto del guadagno dell'am-

plificatore per l'attenuazione della rete di reazione sia maggiore di uno ed inoltre lo sfasamento totale introdotto dai due elementi deve essere pari a n volte 360°.

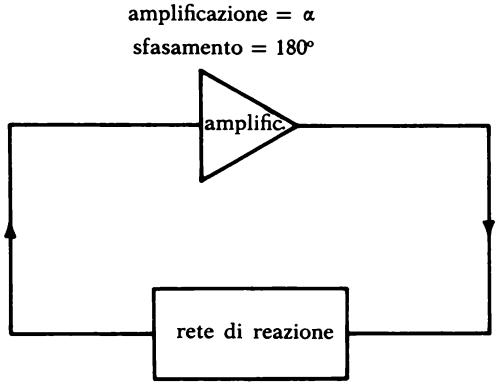


FIG. 3.-
Schema a blocchi di un oscillatore.

Queste condizioni implicano che le oscillazioni avvengano in ogni sistema in cui un segnale amplificato venga riportato in fase all'ingresso dell'amplificatore dopo essere stato attenuato ma in misura minore dell'originaria amplificazione. In un tale sistema la stabilità di frequenza dipende principalmente dai cambiamenti di fase introdotti dalla rete di reazione. Per ottenere elevate stabilità, sono usati comunemente cristalli al quarzo come elementi di reazione.

Un cristallo di quarzo (in realtà è un lamina opportunamente ricavata da un cristallo), può essere schematizzato come un bipolo il cui circuito equivalente è mostrato in figura 4.

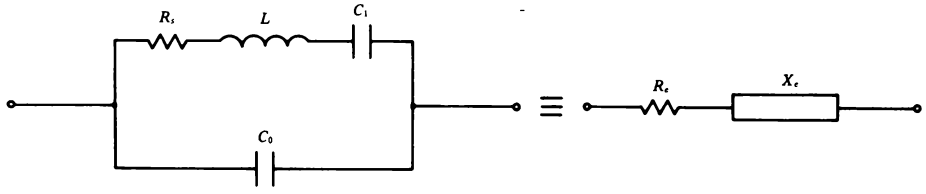


FIG. 4. - Circuito equivalente per un cristallo al quarzo.

In esso, considerando che vibri sotto l'azione di una tensione applicata, R_S corrisponde all'attrito meccanico delle vibrazioni, L corrisponde alla massa, C_1 all'inverso della rigidezza meccanica ed infine C_0 alla capacità tra gli elettrodi.

La tabella 2 mostra i valori tipici dei componenti del circuito equivalen-

te per diversi tipi di taglio e per diverse frequenze.

Il valore della frequenza di risonanza dipende dal modo in cui è stato effettuato il taglio della lamina. Nella figura 5 sono mostrati cristalli ottenuti con tagli X oppure Y. Vengono definiti a taglio X quelli ottenuti da un taglio perpendicolare all'asse X, mentre vengono definiti a taglio Y quelli ottenuti da un taglio perpendicolare all'asse Y. Altri tipi di tagli vengono indicati con AT, DT, NT ecc.

frequenza	32 KHz	280 KHz	525 KHz	2 MHz
taglio	YX	DT	DT	AT
R_s (Ω)	40 K	1820	1400	82
L (H)	4800	25,9	12,7	0,52
C_1 (pF)	0,00491	0,0125	0,00724	0,0122
C_0 (pF)	2,85	5,62	3,44	4,27
Q	25000	25000	30000	80000

Tabella 2. - Valori tipici dei componenti del circuito equivalente di un cristallo al quarzo.

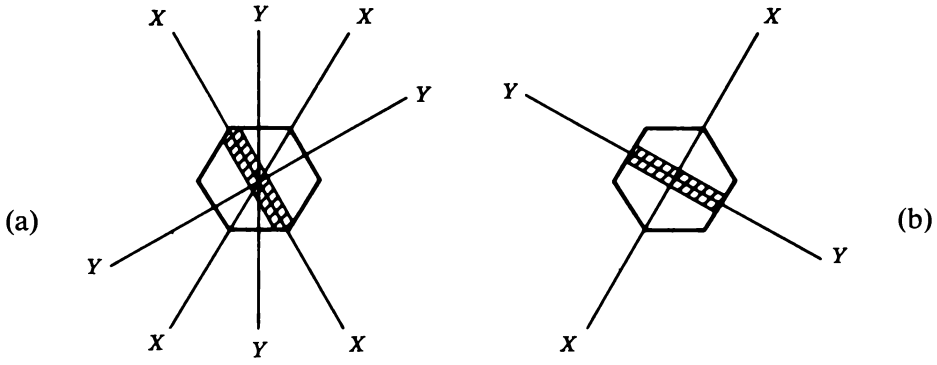


FIG. 5. - Cristalli di quarzo a tagli X (b) ed Y (a).

Il circuito equivalente può essere semplificato in un unico componente resistivo (R_e) ed in uno reattivo (X_e). Quest'ultimo si annulla in corrispondenza di due frequenze di risonanza indicate con: f_R (frequenza di risonanza) ed f_a (frequenza di antirisonanza).

Il ramo composto da R_s, L e C_1 è un circuito risonante serie in corrispondenza ad una frequenza

$$f_r = \frac{1}{2\pi\sqrt{LC}}$$

mentre il parallelo dei due rami oscillerà ad una frequenza intermedia tra f_r ed f_a che dipende dal valore di una capacità posta in parallelo al quarzo come si vedrà più avanti.

Si può dimostrare che:

$$f_a \cong f_r \left(1 + \frac{C}{C_0}\right)^{\frac{1}{2}}$$

dove il rapporto C/C_0 vale circa 0,003, per cui f_a è approssimativamente uguale a 1,0015 f_r. Virtualmente tutti gli oscillatori a cristallo sono progettati per operare tra f_r ed f_a per cui la differenza di frequenze è importante.

Configurazione del circuito di reazione

Un buon amplificatore per la sezione amplificatrice è un inverter CMOS con un resistore di valore elevato connesso fra l'ingresso e l'uscita, come è mostrato nella fig. 6. Il valore di R_f deve essere sufficientemente elevato (maggiore di 10 M Ω) in modo da non influenzare l'attenuazione e la fase della rete di reazione.

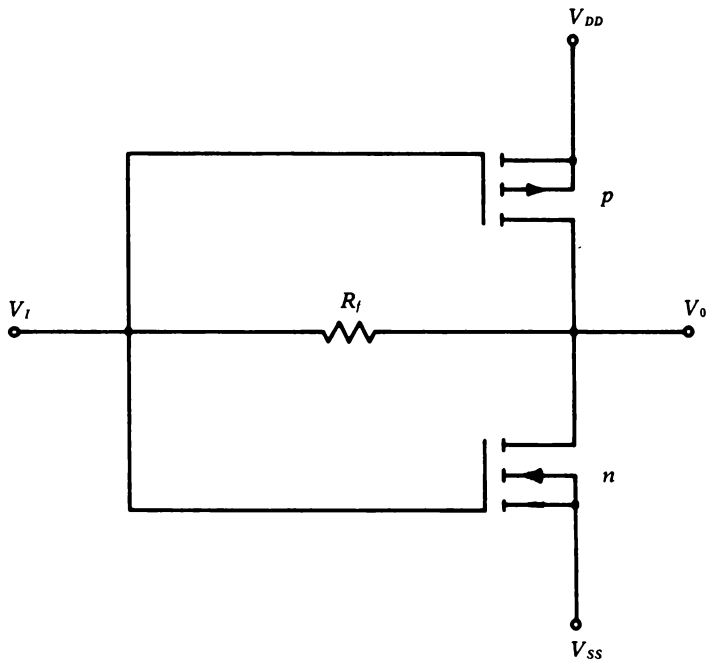


FIG. 6. - Tipico inverter CMOS con resistore di reazione.

Il suo compito è di polarizzare l'uscita alla stessa tensione dell'ingresso ($V_0 = V_{in}$); questo punto è pari a circa la metà della tensione di alimentazione come si può riscontrare dalle caratteristiche di trasferimento di un inverter. Con $V_{DD} = 10V$ una variazione di $\pm$ 1 volt attorno al punto di polarizzazione, causa una variazione della tensione di uscita di circa 10 V. In tal modo il guadagno di tensione (K_A) è pari a circa 5 ed aumenta fino a circa 10 se la tensione di alimentazione scende a 5V, poiché le caratteristiche diventano più squadrate.

Un circuito oscillante, allora, può essere formato utilizzando un inverter ed una rete di reazione avente una attenuazione costante (β) maggiore di 0,2. Nella figura 7 è mostrato un circuito con rete a pi che possiede queste caratteristiche.

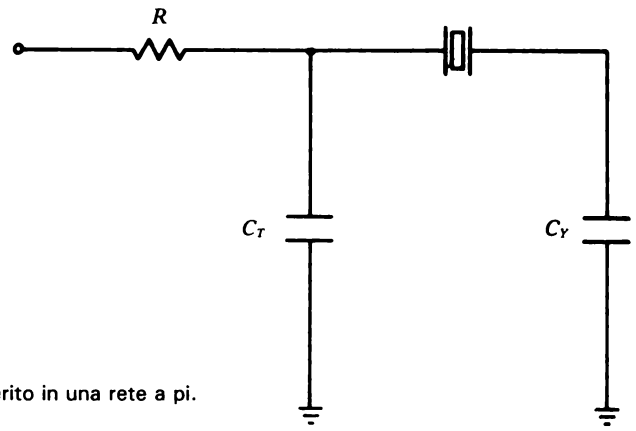


FIG. 7. - Cristallo inserito in una rete a pi.

Un oscillatore può essere progettato con qualsiasi cristallo conoscendo il valore massimo della sua resistenza equivalente e della capacità di carico C_L. Quest'ultima viene inserita in parallelo alla C_0 per ridurre l'effetto delle capacità parassite che farebbero cambiare la frequenza di risonanza. Elevati valori di C_L generalmente migliorano la stabilità in frequenza ma aumentano la dissipazione di potenza; attualmente i migliori valori per C_L rientrano tra 10 e 30pF. La scelta della capacità equivalente totale C_L, fissa solamente la somma delle due poste in serie

$$\frac{C_T \cdot C_Y}{C_T + C_Y}$$

nella rete a pi; i loro valori singoli possono essere trovati mediante le espressioni:

$$C_T = \frac{4C_L}{1 - 5\,f\,R_e\,C_L} \qquad [1]$$

$$C_Y = \frac{4\,C_L}{3 + 5\,f\,R_e\,C_L} \qquad [1]$$

Volendo ottenere valori precisi della frequenza di oscillazione al posto di uno dei due condensatori fissi può essere utilizzato uno variabile.

Circuiti pratici

L'inverter usato come amplificatore, la rete di reazione ed il cristallo possono essere combinati insieme per realizzare un oscillatore. Ad esempio utilizzando il circuito di figura 8 che impiega un CD 4049, supponendo di richiedere una oscillazione a 2 MHz e fissando una $C_L = 20$ pF, i valori di C_T e C_Y possono ricavarsi delle equazioni [1]:

$$C_T \cong 80 \text{ pF}$$
$$C_Y \cong 26 \text{ pF}$$

nelle quali il valore di R_{emax} ricavato dalle caratteristiche del cristallo vale circa 100 Ω, ed a C_Y deve essere sottratto il valore della capacità di ingresso dell'amplificatore.

Il valore della resistenza R del circuito di reazione può essere calcolato nel modo seguente:

$$R = \frac{(3X_e + 0{,}27\,R_e)\,(X_e - 0{,}8R_e)}{16\,R_e} \cong 30 \text{ KΩ}$$

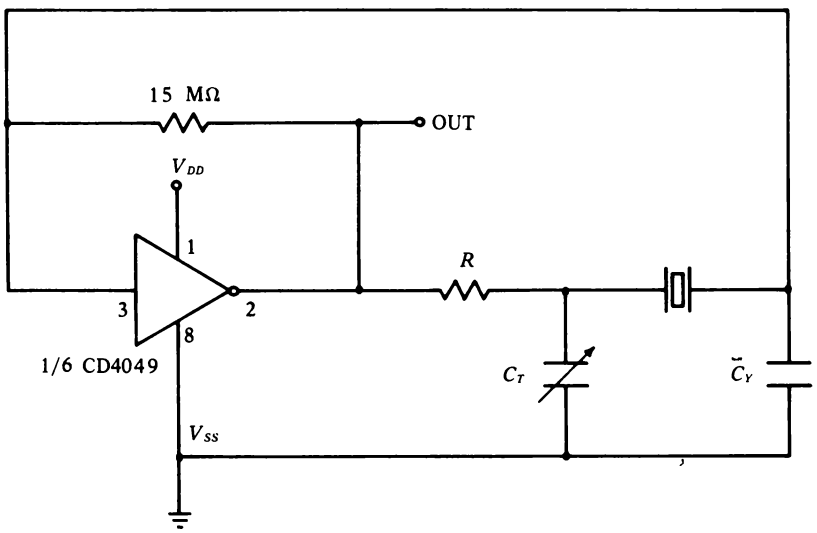


FIG. 8. - Clock controllato al cristallo, impiegante un CMOS CD 4049.

dove $X_e = X_{CL} \cong 4K\ \Omega$.

Nei circuiti impieganti CMOS si possono ottenere frequenze di lavoro da pochi KHz ad oltre 10MHz, tenendo presente però che alle più basse frequenze è necessario connettere in parallelo più inverter per fornire una più elevata corrente di pilotaggio al cristallo.

Nel caso che non si voglia ricorrere alla diretta progettazione del circuito di clock, molti produttori di μP mettono a disposizione innumerevoli circuiti monolitici come ad esempio l'8224 ed il K 1117 per l'8080, l'MC6870 per il 6800 e l'F 4702 per lo Z 80.

4. Il bus e la consolle

La realizzazione di una struttura modulare intercambiabile, richiede l'adozione di una precisa definizione dei parametri fisici e funzionali. A tale scopo sono possibili due tipi di approccio: l'adozione di uno standard, sia per il bus che per le dimensioni, del tipo di quelli descritti nel capitolo quinto oppure, per sistemi di ridotte dimensioni, la creazione di un sistema personalizzato.

Nel caso presente si è optato per la seconda soluzione ed in particolare sono state adottate schede di formato Eurocard (100×160 mm) con connettori da 32 piedini per faccia. La figura 9 mostra la loro disposizione, mentre nella tabella 3 ne viene data una succinta descrizione funzionale che, peraltro, verrà ulteriormente chiarita nel corso del progetto.

Senza entrare in dettaglio si può osservare che sono presenti:
- 2 linee per l'alimentazione;
- 1 linea per il clock;
- 16 linee per gli indirizzi;
- 8 linee per i dati;
- 24 linee per i controlli;
- 13 linee a disposizione per eventuali espansioni e variazioni.

Nella figura 10 è invece mostrata la struttura della consolle di tipo hardware utilizzata per l'interazione tra il sistema e l'operatore. Questo tipo di consolle, sebbene meno potente di quelle di tipo software (viene chiamata consolle software quella che utilizza un programma per la sua lettura), possiede alcune particolarità che si rivelano molto utili, quali ad esempio il controllo diretto delle linee del bus, per una più agevole comprensione del funzionamento del sistema. Essa è composta da:
- 16 deviatori per l'impostazione degli indirizzi
- 8 deviatori per l'impostazione dei dati
- 7 deviatori per i controlli

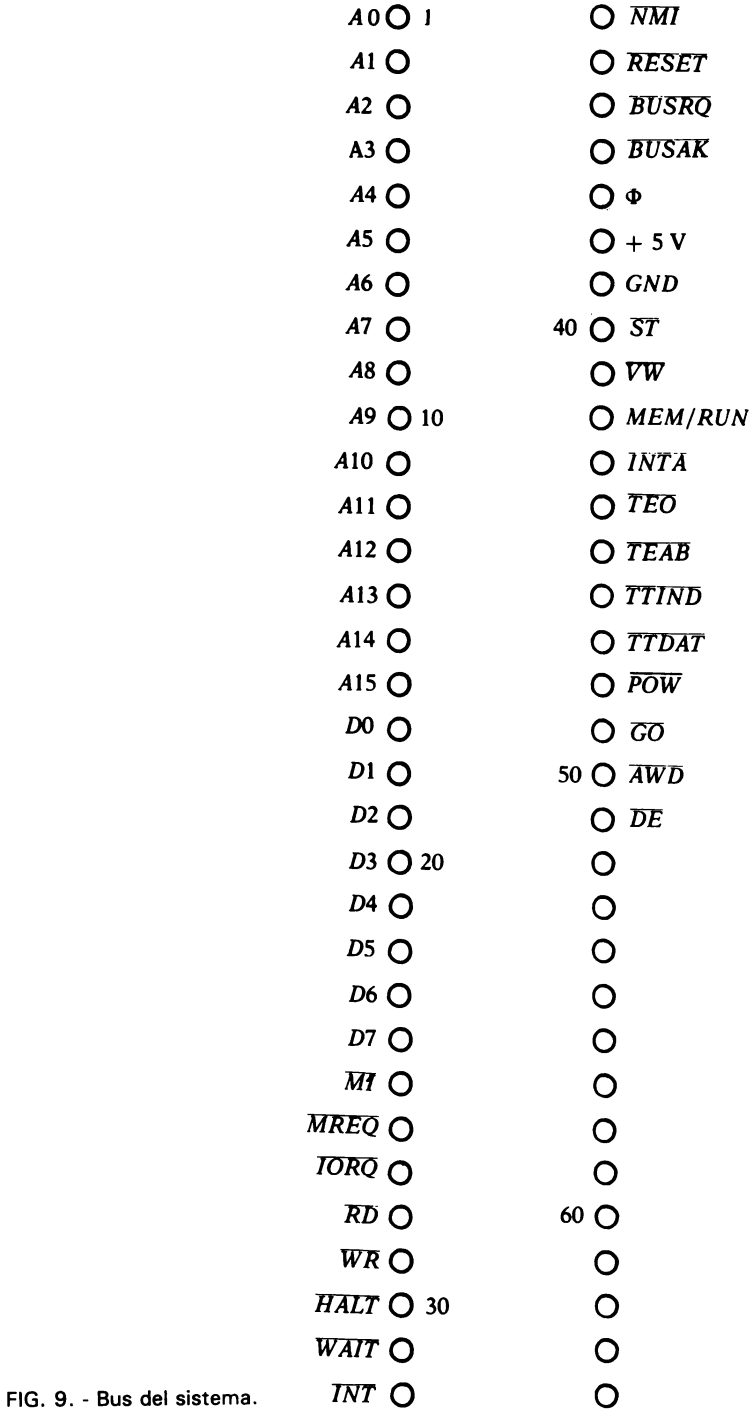


FIG. 9. - Bus del sistema.

piedino	nome	descrizione
1-16	indirizzi	linee di indirizzo
17-24	dati	linee dei dati
25	$\overline{M1}$	ciclo macchina uno
26	$\overline{MREQ}$	richiesta di memoria
27	$\overline{IORQ}$	richiesta di I/0
28	$\overline{RD}$	lettura di memoria o dispositivo esterno
29	$\overline{WR}$	scrittura in memoria o dispositivo esterno
30	$\overline{HALT}$	stato di halt
31	$\overline{WAIT}$	stato di attesa
32	$\overline{INT}$	richiesta di interruzione
33	$\overline{NMI}$	richiesta di interruzione non mascherabile
34	$\overline{RESET}$	reset manuale del sistema
35	$\overline{BUSRQ}$	richiesta del bus
36	$\overline{BUSAK}$	riconoscimento del bus
37	Φ	clock del sistema
38	+ 5 V	alimentazione a + 5 V continua
39	GND	massa digitale
40	$\overline{ST}$	comando di memorizzazione (Store)
41	$\overline{VW}$	comando di abilitazione del display (View)
42	MEM/RUN	comando memorizzazione o esecuzione del programma (Memory/Running)
43	$\overline{INTA}$	riconoscimento interruzione
44	$\overline{TEO}$	abilitazione di uscita (Tri-state Enable Out)
45	$\overline{TEAB}$	abilitazione tri-state esterni
46	$\overline{TTIND}$	abilitazione tri-state indirizzi
47	$\overline{TTDAT}$	abilitazione tri-state dei dati
48	$\overline{POW}$	reset automatico all'accensione
49	$\overline{GO}$	comando inizio esecuzione del programma
50	$\overline{AWD}$	abilitazione memoria e display
51	$\overline{DE}$	abilitazione eventuale dispositivo esterno

Tabella 3. - Bus del sistema, descrizione funzionale.

- 5 deviatori a disposizione e non connessi
- 4 display a sette segmenti per gli indirizzi
- 2 display a sette segmenti per i dati.

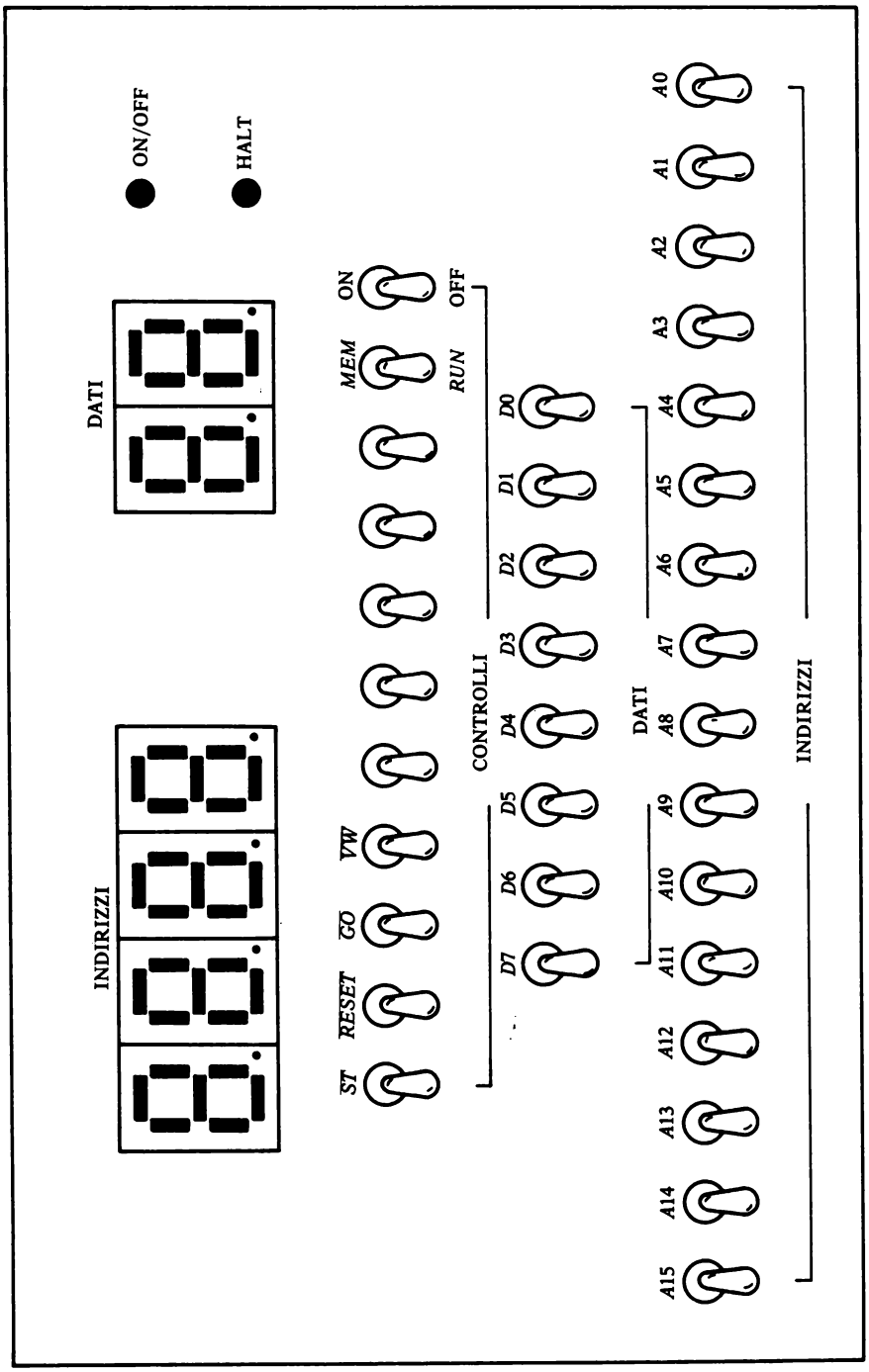


FIG. 10. - Consolle hardware del microcomputer.

5. La memoria e la sua abilitazione

La memoria RAM utilizzata è la MCM 68B10 della Motorola che è di tipo statico ed organizzata in 128 bytes di 8 bit ciascuno. In figura 11 e seguenti, sono mostrate le sue caratteristiche fisiche ed elettriche. Dallo schema a blocchi di figura 12 si rileva che per l'abilitazione della memoria, i piedini CS0 ÷ CS5 debbono presentare la seguente configurazione: 100100. Nell'attuale progetto non sono state utilizzate tutte le sei possibilità ma solo quella relativa a CS0, mentre le altre vengono abilitate in modo stabile mediante connessione a massa o a + V_{cc}.

Un'altra caratteristica rilevabile dalla figura è la presenza di un bus bidirezionale comandato dal piedino 16 (Read/Write).

Affinchè avvenga una operazione di lettura, sull'address bus deve essere presente l'indirizzo della locazione interessata ed inoltre i pin CS (con CS vengono indicati quelli attivi al livello alto = 1) devono portarsi al livello alto, mentre i pin $\overline{CS}$ (quelli che sono attivi al livello logico zero) devono portarsi al livello basso. A questo punto, è il livello presente sul pin R/W che seleziona l'operazione di lettura o di scrittura.

Nelle temporizzazioni rivestono particolare importanza i due parametri:

- tempo di accesso t_{acc}
- tempo di ciclo t_{cyc}

Il primo parte dall'istante in cui è stabile sull'address bus l'indirizzo della locazione desiderata e termina all'istante in cui il dato valido è presente sul bus dei dati. Dalla tabella si può notare che esso ha un valore massimo pari a 250 ns. Il tempo di ciclo è invece quello che intercorre da quando sono stabili gli indirizzi a quando è conclusa l'operazione sulla locazione di memoria interessata e ne può iniziare una successiva.

Un altro parametro che occorre tener presente è il t_{RCS}. Questo indica il tempo intercorrente da quanto è stabile, sul piedino READ il valore attivo (secondo l'operazione che si intende effettuare), a quando le abilitazioni sono nella loro configurazione di funzionamento. Il suo valore minimo può essere zero, ciò significa che i segnali sul pin R/W e sulle abilitazioni possono arrivare contemporaneamente.

Nella figura 14 è illustrata la temporizzazione di un ciclo di scrittura il cui funzionamento è analogo a quello precedente. I tempi t_{cyc} (W) e t_{WCS} sono i corrispondenti di t_{cyc} (R) e di t_{RCS}.

Gli altri parametri sono intuibili facilmente dai diagrammi.

Rete di abilitazione della memoria

In figura 15 è mostrata la rete di abilitazione della memoria che prevede tre possibilità diverse:

1° caso - $\overline{BUSAK}$ attivo, $\overline{ST}$ attivo,

Questi valori indicano che la CPU è nello stato di alta impedenza ($\overline{BU}$-

MOTOROLA

MCM6810
1.0 MHz
MCM68A10
1.5 MHz
MCM68B10
2.0 MHz

MOS
(N-CHANNEL, SILICON-GATE)
128 X 8-BIT STATIC RANDOM ACCESS MEMORY

128 X 8-BIT STATIC RANDOM ACCESS MEMORY

The MCM6810 is a byte-organized memory designed for use in bus-organized systems. It is fabricated with N-channel silicon-gate technology. For ease of use, the device operates from a single power supply, has compatibility with TTL and DTL, and needs no clocks or refreshing because of static operation.

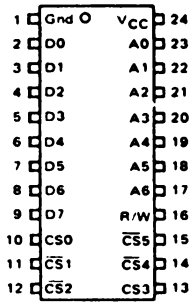
The memory is compatible with the M6800 Microcomputer Family, providing random storage in byte increments. Memory expansion is provided through multiple Chip Select inputs.

- Organized as 128 Bytes of 8 Bits
- Static Operation
- Bidirectional Three-State Data Input/Qutput
- Six Chip Select Inputs (Four Active Low, Two Active High)
- Single 5-Volt Power Supply
- TTL Compatible
- Maximum Access Time = 450 ns – MCM6810
  360 ns – MCM68A10
  250 ns – MCM68B10

P SUFFIX
PLASTIC PACKAGE
CASE 709

L SUFFIX
CERAMIC PACKAGE
CASE 716

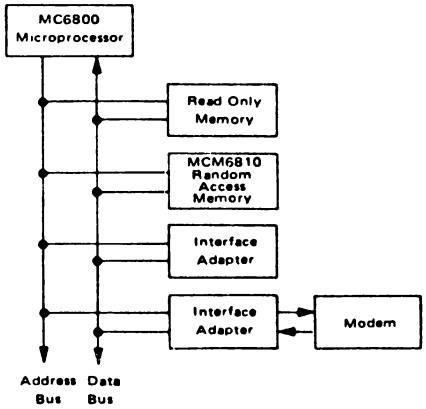
PIN ASSIGNMENT



ORDERING INFORMATION

Speed	Device	Temperature Range
1.0 MHz	MC6810P, L	0 to 70°C
	MC6810CP, CL	-40 to +85°C
MIL-STD-883B	MC6810BJCS	-55 to +125°C
MIL-STD-883C	MC6810CJCS	
1.5 MHz	MC68A10P, L	0 to +70°C
	MC68A10CP, CL	-40 to +85°C
2.0 MHz	MC68B10P, L	0 to +70°C

M6800 MICROCOMPUTER FAMILY
BLOCK DIAGRAM



MCM6810 – RANDOM ACCESS MEMORY
BLOCK DIAGRAM

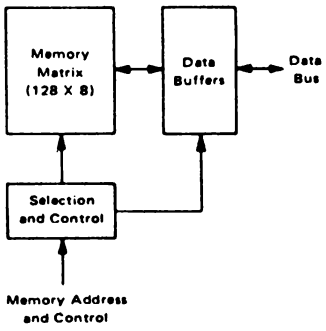


FIG. 11. - Piedinatura della RAM MCM 6810 (Motorola).

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	Vdc
Input Voltage	V_{in}	-0.3 to +7.0	Vdc
Operating Temperature Range	T_A	T_L to T_H 0 to 70 -40 to 85 -55 to 125	°C
Storage Temperature Range	T_{stg}	-65 to +150	°C
Thermal Resistance	θ_{JA}	82.5	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit.

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 V ±5%, V_{SS} = 0, T_A = T_L to T_H unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
Input Current (A_n, R/W, CS_n, $\overline{CS_n}$) (V_{in} = 0 to 5.25 V)	I_{in}	-	-	2.5	μAdc
Output High Voltage (I_{OH} = -205 μA)	V_{OH}	2.4		-	Vdc
Output Low Voltage (I_{OL} = 1.6 mA)	V_{OL}	-	-	0.4	Vdc
Output Leakage Current (Three-State) (CS = 0.8 V or $\overline{CS}$ = 2.0 V, V_{out} = 0.4 V to 2.4 V)	I_{TSI}		-	10	μAdc
Supply Current 1.0 MHz (V_{CC} = 5.25 V, all other pins grounded) 1.5, 2.0 MHz	I_{CC}	-		80 100	mAdc
Input Capacitance (A_n, R/$\overline{W}$, CS_n, $\overline{CS_n}$) (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)	C_{in}	-	-	7.5	pF
Output Capacitance (D_n) (V_{out} = 0, T_A = 25°C, f = 1.0 MHz, CSØ = 0)	C_{out}	-		12.5	pF

RECOMMENDED DC OPERATING CONDITIONS

Parameter	Symbol	Min	Nom	Max	Unit
Input High Voltage	V_{IH}	2.0	-	5.25	Vdc
Input Low Voltage	V_{IL}	-0.3	-	0.8	Vdc

BLOCK DIAGRAM

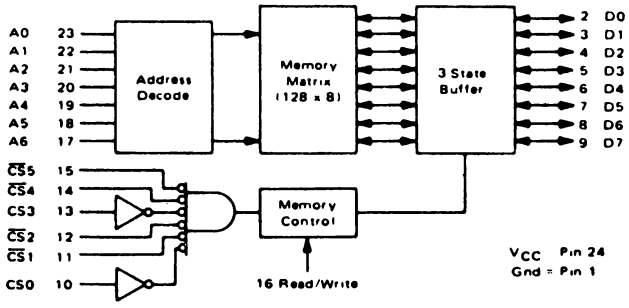
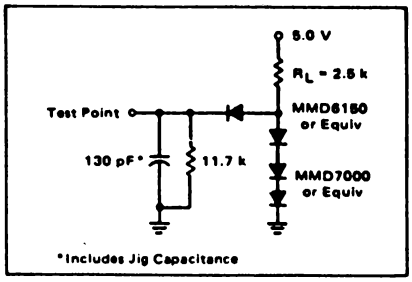


FIG. 12. - RAM 6810 (Motorola).

AC TEST CONDITIONS

Condition	Value
Input Pulse Levels	0.8 V to 2.0 V
Input Rise and Fall Times	20 ns
Output Load	See Figure 1

FIGURE 1 – AC TEST LOAD

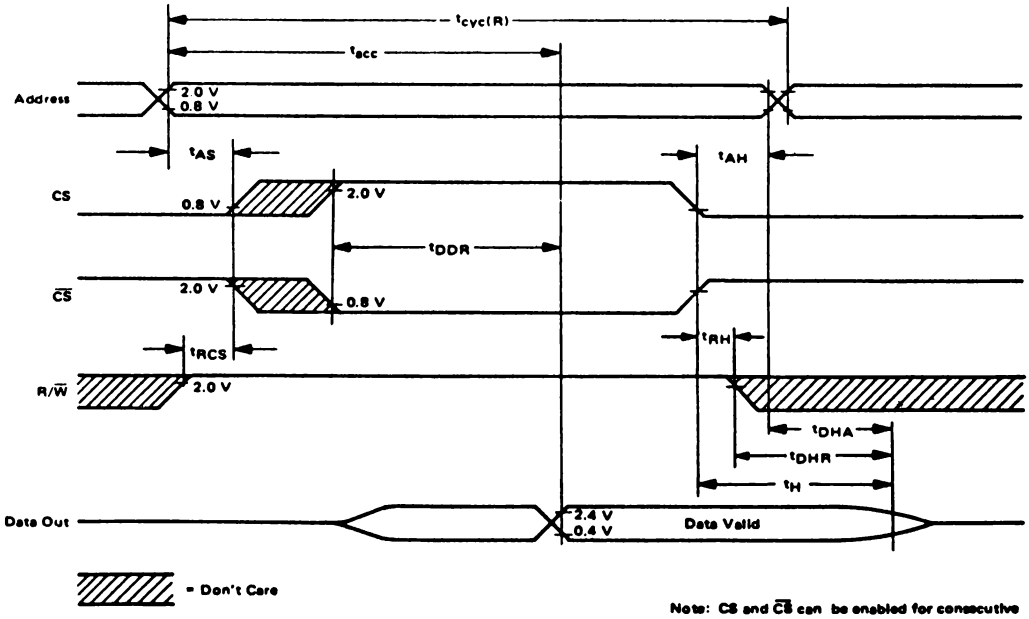


AC OPERATING CONDITIONS AND CHARACTERISTICS

READ CYCLE (V_{CC} = 5.0 V ±5%, V_{SS} = 0, T_A = T_L to T_H unless otherwise noted.)

Characteristic	Symbol	MCM6810		MCM68A10		MCM68B10		Unit
		Min	Max	Min	Max	Min	Max	
Read Cycle Time	$t_{cyc(R)}$	450	–	360	–	250	–	ns
Access Time	t_{acc}	–	450	–	360	–	250	ns
Address Setup Time	t_{AS}	20	–	20	–	20	–	ns
Address Hold Time	t_{AH}	0	–	0	–	0	–	ns
Data Delay Time (Read)	t_{DDR}	–	230	–	220	–	180	ns
Read to Select Delay Time	t_{RCS}	0	–	0	–	0	–	ns
Data Hold from Address	t_{DHA}	10	–	10	–	10	–	ns
Output Hold Time	t_H	10	–	10	–	10	–	ns
Data Hold from Read	t_{DHR}	10	80	10	60	10	60	ns
Read Hold from Chip Select	t_{RH}	0	–	0	–	0	–	ns

READ CYCLE TIMING



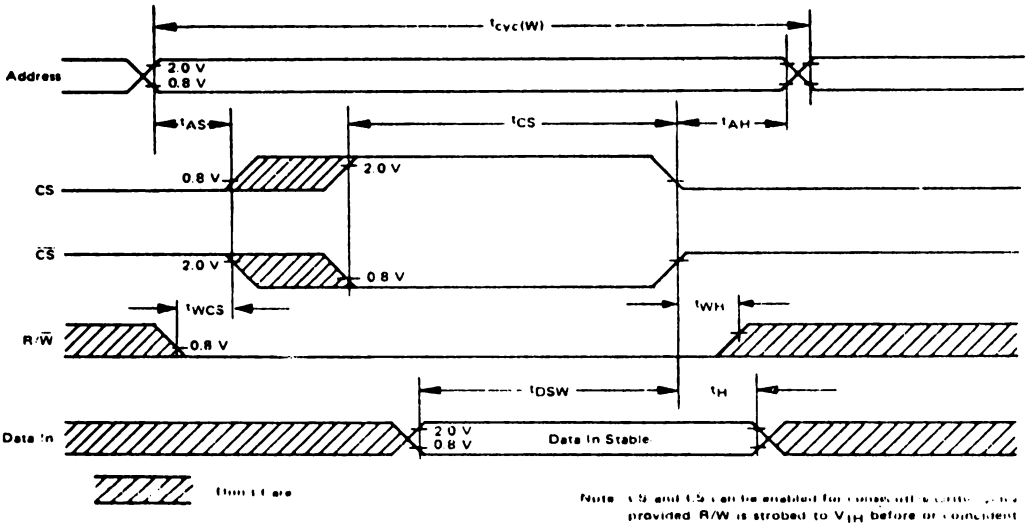
Note: CS and $\overline{CS}$ can be enabled for consecutive read cycles provided R/W remains at V_{IH}.

FIG. 13. - RAM 6810 (Motorola).

WRITE CYCLE (V_{CC} = 5.0 V ± 5%, V_{SS} = 0, T_A = T_L to T_H unless otherwise noted)

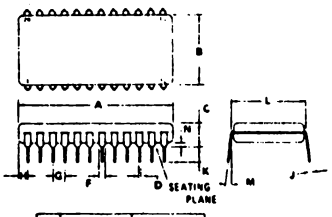
Characteristic	Symbol	MCM6810		MCM68A10		MCM68B10		Unit
		Min	Max	Min	Max	Min	Max	
Write Cycle Time	$t_{cyc(W)}$	450	–	360	–	250	–	ns
Address Setup Time	t_{AS}	20	–	20	–	20	–	ns
Address Hold Time	t_{AH}	0	–	0	–	0	–	ns
Chip Select Pulse Width	t_{CS}	300	–	250	–	210	–	ns
Write to Chip Select Delay Time	t_{WCS}	0	–	0	–	0	–	ns
Data Setup Time (Write)	t_{DSW}	190	–	80	–	60	–	ns
Input Hold Time	t_H	10	–	10	–	10	–	ns
Write Hold Time from Chip Select	t_{WH}	0	–					

WRITE CYCLE TIMING



Note ... provided R/W is strobed to V_{IH} before or coincident with the Address change, and remains high for time t_{AS}.

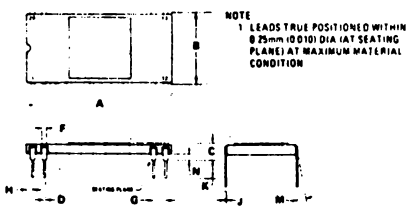
PACKAGE DIMENSIONS



DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	31.37	32.13	1.235	1.265
B	13.72	14.22	0.540	0.560
C	4.57	5.08	0.180	0.200
D	0.36	0.51	0.014	0.020
F	1.02	1.32	0.040	0.060
G	2.41	2.67	0.095	0.105
H	1.78	2.03	0.070	0.080
J	0.20	0.30	0.008	0.012
K	3.05	3.56	0.120	0.140
L	14.73	15.24	0.580	0.600
M	0°	10°	0°	10°
N	0.51	1.02	0.020	0.040

NOTES
1. LEADS TRUE POSITIONED WITHIN 0.25 mm (0.010) DIA AT SEATING PLANE AT MAXIMUM MATERIAL CONDITION (DIM "D")
2. DIM "L" TO CENTER OF LEADS WHEN FORMED PARALLEL

CASE 709-01
(PLASTIC)



DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	29.97	30.99	1.180	1.220
B	14.86	15.62	0.585	0.615
C	3.05	4.19	0.120	0.165
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.30	0.008	0.012
K	2.54	4.19	0.100	0.165
M		10°		10°
N	0.51	1.52	0.020	0.060

CASE 716-02
(CERAMIC)

FIG. 14. - RAM 6810 (Motorola).

$\overline{SAK}$ attivo) e che da tastiera si vuole compiere una operazione di scrittura in memoria ($\overline{ST}$ attivo). Si tenga presente che la memoria viene abilitata con un livello logico alto, da qui la presenza del NOR).

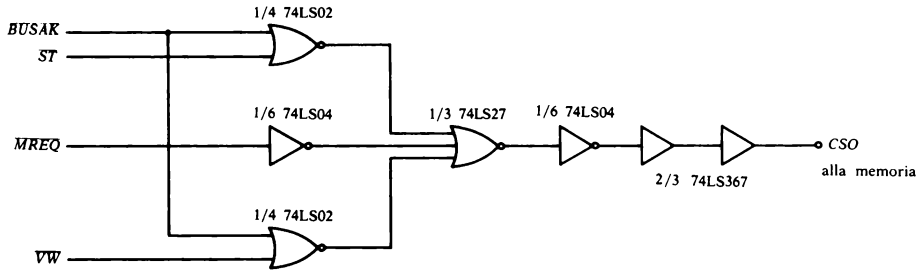


FIG. 15. - Rete per l'abilitazione della memoria.

2° caso - $\overline{MREQ}$ attivo.

Questa condizione esprime una richiesta da parte della CPU di colloquiare con la memoria. Poiché $\overline{MREQ}$ è attivo basso, viene semplicemente complementato.

3° caso - $\overline{BUSAK}$ attivo, $\overline{VW}$ attivo.

In questa situazione la CPU è in uno stato di alta impedenza e da tastiera, tramite il comando $\overline{VW}$, viene richiesta una operazione di lettura in memoria.

La presenza del NOT e dei buffer all'uscita del triplo NOR deriva dalla considerazione che il segnale CSO deve arrivare alla memoria dopo che sia giunto il segnale R/W. La condizione limite è data dalla contemporaneità dei due segnali infatti, come si è visto in precedenza, i tempi t_{RCS} e t_{WCS} potevano essere uguali a zero.

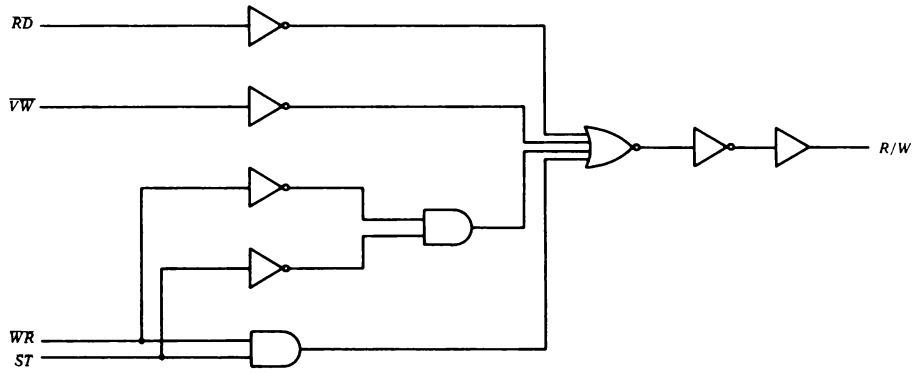
Per verificare il rispetto di questa condizione, nella seguente tabella è riportato il calcolo dei ritardi della rete di abilitazione e di quella pilotante il pin R/W che è riportata in figura 16.

Ritardo rete abilitazione		ritardo rete R/W	
porta	tempo di prop.	porta	tempo di prop.
NOR 7402	10 ns	AND 7408	14 ns
NOR 7427	8 ns	NOR 7432	12 ns
NOT 7404	10 ns	BUFFER 74LS367	10 ns
BUFFER 74LS367	10×2 ns		
TOTALE	48 ns	TOTALE	36 ns

Da essa si può osservare che il segnale di abilitazione arriva dopo 12 ns rispetto al comando di R/W permettendo di operare in condizioni di sufficiente sicurezza.

Rete di comando lettura/scrittura (R/W)

Nella figura 16 è mostrata la rete per il comando di lettura o scrittura in memoria insieme alla sua tabella della verità. In essa compaiono le quattro configurazioni dei segnali di ingresso $\overline{VW}$, $\overline{RD}$, $\overline{WR}$ e $\overline{ST}$ per le quali la



$\overline{VW}$	$\overline{RD}$	$\overline{WR}$	$\overline{ST}$	U
0	1	1	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	0	0

FIG. 16. - Rete di comando del pin R/W.

memoria deve essere abilitata o per una operazione di scrittura (U = 0 logico), oppure per una di lettura (U = 1 logico). Le quattro righe della tabella possono spiegarsi nel modo seguente:
1° caso - Questa configurazione, in cui è attivo il solo segnale $\overline{VW}$, indica che dalla tastiera è stato inviato un comando di lettura della memoria;
2° caso - indica una richiesta di lettura in memoria da parte della CPU;
3° caso - indica una richiesta di scrittura in memoria da parte della CPU;
4° caso - indica che dalla tastiera, tramite il comando $\overline{ST}$, è stato inviato un ordine di scrittura.

La rete di figura 16 è stata ottenuta mediante la semplificazione tramite le mappe di Karnaugh, nelle quali sono state considerate condizioni indifferenti tutte quelle configurazioni in cui più di un segnale in ingresso è attivo (0 logico). Tali configurazioni non si possono verificare, infatti sono relative a tre diverse possibilità:
a) che sia attivo un segnale da tastiera ed uno dalla CPU, ma ciò è impossibile poiché l'abilitazione del tri-state della consolle impone che la CPU si trovi in uno stato di alta impedenza, vale a dire che una operazione sulla memoria effettuata tramite consolle pone automaticamente basso il pin $\overline{BUSRQ}$ della CPU;
b) che siano attivi simultaneamente i due segnali $\overline{RD}$ e $\overline{WR}$, ma anche questa operazione è impossibile poiché non esiste alcun codice operativo che richieda tale stato.
c) che siano attivati contemporaneamente sulla consolle due controlli. È questa l'unica condizione di rischio poiché se un operatore dovesse premere contemporaneamente i tasti ST e VW, potrebbe verificarsi in memoria una operazione non desiderata. Ma, all'atto della semplificazione, le condizioni di indifferenza sono state prese come 1 logico per cui la rete logica darebbe origine ad una operazione di lettura che non modifica la memoria.

6. I comandi della consolle

La sezione dati ed indirizzi della consolle hardware è collegata ai bus principali mediante dispositivi tri-state, in modo che la consolle stessa possa essere trattata dal sistema come un dispositivo esterno. Infatti, in particolari situazioni, i 24 deviatori si trovano in uno stato di alta impedenza risultando virtualmente sconnessi dal resto del sistema.

sezione dati

È costituita da otto deviatori e da otto buffer tri-state 74LS367 le cui uscite sono collegate al bus dei dati (DO ÷ D7)

L'abilitazione, comune a tutti gli otto buffer, avviene tramite il controllo $\overline{TTDAT}$, implementato dalla rete logica di figura 17.

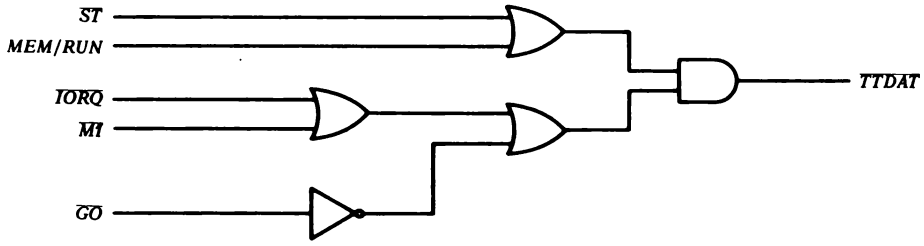


FIG. 17. - Rete logica per l'abilitazione della sezione dati della consolle.

Quest'ultimo diventa attivo (livello logico basso), abilitando i tri-state a trasferire sul bus il byte impostato tramite gli otto deviatori, in due precise configurazioni degli ingressi:
1° caso - dalla consolle viene richiesta una operazione di scrittura in memoria del dato posto sulle otto linee del bus dati. Ciò avviene portando i deviatori $\overline{\text{MEM/RUN}}$ e $\overline{\text{ST}}$ al livello logico 0.
2° caso - dalla consolle si richiede una interruzione mediante il pulsante $\overline{\text{GO}}$. La CPU risponde a questo segnale tramite la attivazione del segnale $\overline{\text{INTA}}$ ottenuto dall'OR di $\overline{\text{IORQ}}$ con $\overline{\text{M1}}$.

Ciò è utile poiché la CPU legge dal bus dei dati un codice operativo che può essere una RESTART contenente l'indirizzo di memoria iniziale del programma da eseguire se non si desidera utilizzare il RESET.

sezione indirizzi

Questa sezione è costituita da 16 deviatori anch'essi collegati tramite buffer tri-state, alle 16 linee del bus degli indirizzi.

L'abilitazione dei tri-state avviene tramite l'unico segnale MEM/RUN che è attivo al livello logico basso.

sezione controlli

I controlli sono effettuati mediante quattro pulsanti ($\overline{\text{ST}}$, $\overline{\text{RESET}}$, $\overline{\text{GO}}$ e $\overline{\text{VW}}$) ed un deviatore (MEM/RUN) le cui funzioni sono le seguenti:

$\overline{\text{ST}}$ - La sua attivazione abilita la memoria a scrivere, nella locazione individuata dai 16 deviatori di indirizzo, il dato impostato sui deviatori dei dati. Contemporaneamente sui display dati ed indirizzi vengono mostrati i rispettivi valori.

$\overline{\text{RESET}}$ - Questo pulsante agisce direttamente sul pin omonimo della CPU Z 80.

$\overline{\text{GO}}$ - Questo comando può essere usato per dare inizio all'esecuzione di un programma già memorizzato. Infatti, esso agisce direttamente sul pin $\overline{\text{INT}}$ della CPU che, se opera in modo zero, attende un codice operativo ad un byte che dà inizio all'esecuzione del programma.

$\overline{\text{VW}}$ - La sua funzione, analoga a quella di $\overline{\text{ST}}$, è quella di abilitare la memoria ad una operazione di lettura. Inoltre sui rispettivi display viene mostrato sia il dato che il suo indirizzo.

MEM-/RUN - È un deviatore che prevede due posizioni: MEM e RUN. Nella posizione MEM (livello 0) ha la duplice funzione di porre in alta impendenza le linee indirizzi e dati della CPU poiché agisce sul suo comando $\overline{\text{BUSRQ}}$ e di collegare i 16 + 8 deviatori (indirizzi + dati) ai rispettivi bus.

Nella posizione RUN, invece, si ottiene una configurazione inversa, vale a dire che la CPU è collegata ai bus mentre sono virtualmente sconnessi i 24 deviatori. Solo nel caso in cui venga premuto il tasto $\overline{GO}$, non avviene la disabilitazione del tri-state per permettere la lettura del codice operativo dalla consolle.

sezione display

I quattro display degli indirizzi (ad es. TIL 322), sono direttamente collegati, tramite quattro decoder 9368, alle sedici linee A0 ÷ A15. Ciò permette di ottenere la visualizzazione sia in fase di lettura della memoria ($\overline{VW}$ attivo), che in fase di scrittura ($\overline{ST}$ attivo). Inoltre, sempre a causa del collegamento diretto, ogni variazione dei 16 deviatori viene direttamente mostrata sul display.

Il display dei dati è abilitato, tramite la rete di figura 18, da tre diverse configurazioni:

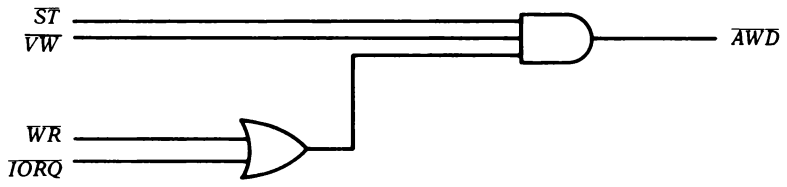


FIG. 18. - Rete per l'abilitazione del display dei dati.

a) attraverso l'esecuzione, da parte della CPU, dell'istruzione OUT che attiva contemporaneamente i segnali $\overline{WR}$ ed $\overline{IORQ}$. In questo caso viene mostrato il dato che viaggiava in quell'istante sul bus dei dati.

b) premendo il tasto $\overline{VW}$. In tal caso viene visualizzato il dato letto dalla memoria.

c) premendo il pulsante $\overline{ST}$. In questo modo si legge, in esadecimale il dato impostato in binario sulle otto linee dati della consolle.

Come risulta dalla figura 19, il circuito completo per il comando del display contiene anche un elemento di memoria per almeno otto bit in parallelo. Ciò permette il mantenimento sul display, del dato che fluisce per pochi istanti sul bus. Infatti ad esempio, nel caso dell'esecuzione dell'istruzione OUT, il tempo di mantenimento è di circa tre cicli di clock.

La memoria, che è costantemente abilitata alla lettura, passa alla fase di scrittura solo al sopraggiungere del segnale $\overline{AWD}$ uscente dalla rete di figura 18, per ritornare poi ad essere abilitata alla lettura.

È inoltre interessante rilevare che non sussistono problemi di temporizzazione poiché il dato da memorizzare è già stabile al sopraggiungere del comando AWD.

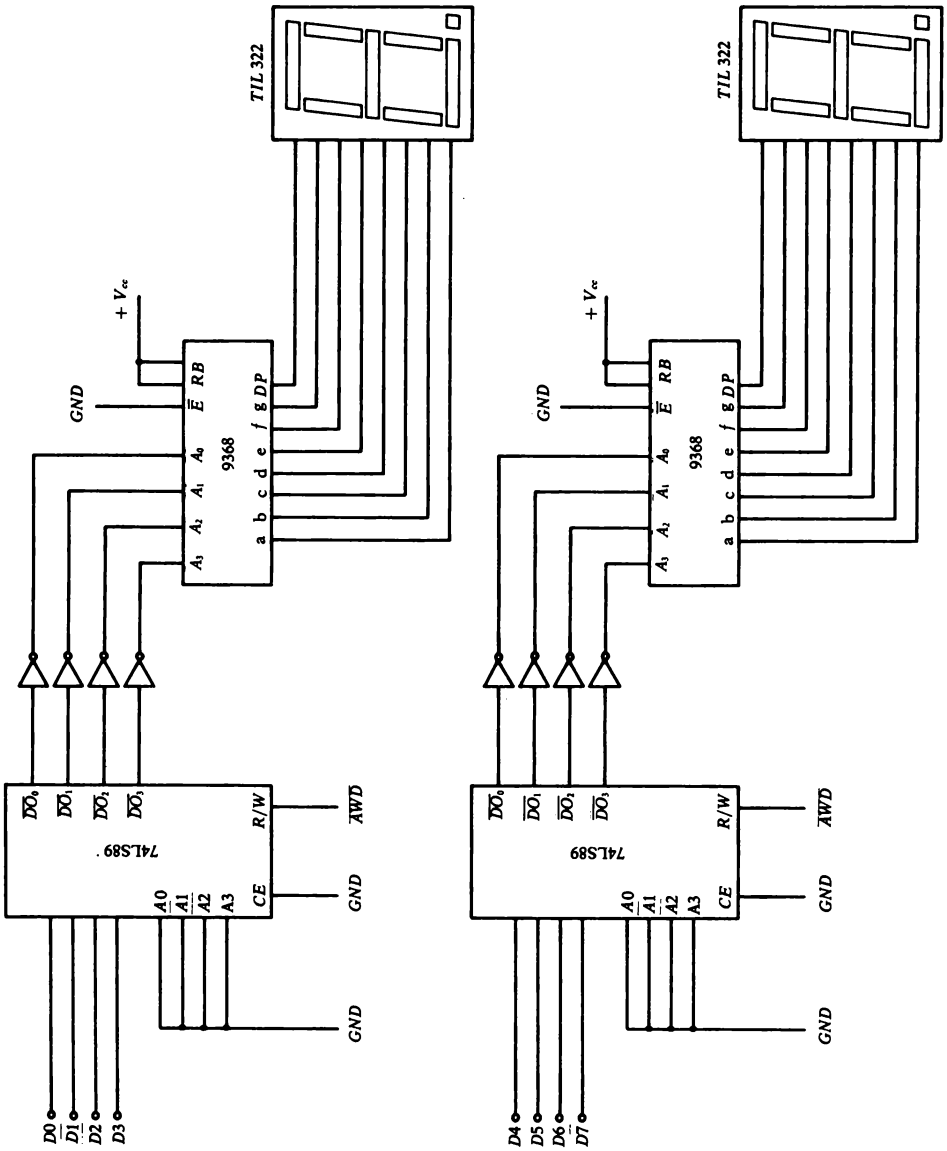


FIG. 19. - Sezione display.

7. Dispositivo esterno

Volendo prevedere l'interfacciamento con un dispositivo esterno, il bus del sistema è stato dotato di una linea denominata $\overline{DE}$. Essa permette, al dispositivo esterno, di generare una interruzione e di fornire o un codice operativo o la parte bassa di un indirizzo se la CPU è stata predisposta per lavorare nel modo 0, 1 o 2 di interruzione.

Coerentemente con la logica del sistema, anche il dispositivo esterno deve essere fornito di tri-state che permette di limitare il numero di congegni collegati in modo contemporaneo alla CPU. A tale scopo si sono create sul bus due linee chiamate $\overline{TEAB}$ e $\overline{TEO}$, che abilitano il buffer solo allorché la CPU riconosce la richiesta di un dispositivo esterno. La rete logica che gestisce questi segnali è mostrata nella figura 20.

Poiché si richiede al bus di essere bidirezionale, ciascuna delle due direzioni del flusso è soggetta ad una abilitazione. Se deve essere effettuato un input verso la CPU sarà attivo $\overline{TEAB}$, mentre sarà attivo $\overline{TEO}$ qualora

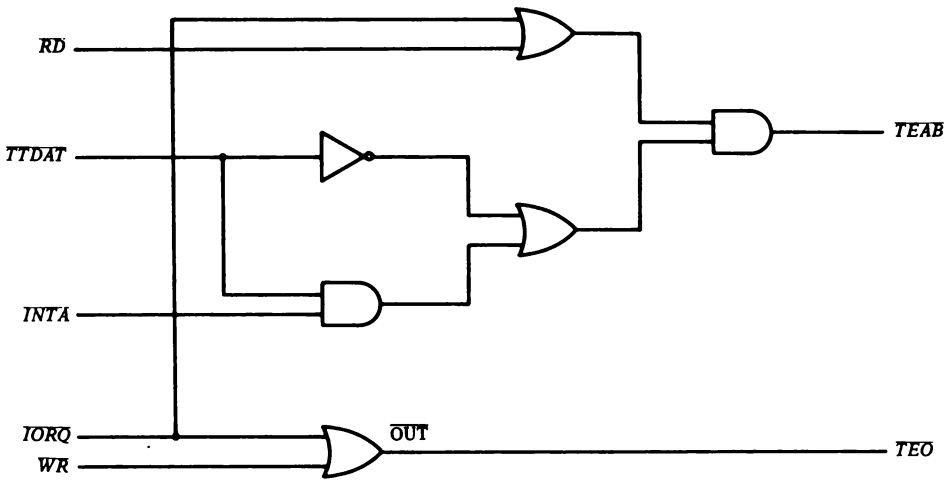


FIG. 20. - Rete per l'abilitazione del dispositivo esterno.

debba essere effettuato un trasferimento verso il dispositivo. La rete mette in evidenza le due possibilità di riconoscimento del dispositivo esterno. La prima è data dalla richiesta di interruzione attraverso $\overline{DE}$, mentre la seconda avviene attraverso le istruzioni di IN ed OUT. Come si osserva, $\overline{TEO}$ è attivato solamente da una operazione di OUT mentre TEAB è attivo sia quando viene eseguita una operazione di IN che quando viene riconosciuta la richiesta di interruzione inviata mediante DE ($\overline{INTA}$).

Può sorgere il problema che, anziché dalla tastiera venga richiesto un byte tramite una istruzione. Per distinguere questi due casi si è inserito nella rete logica di figura 20 il segnale di abilitazione del tri-state delle otto linee della consolle ($\overline{TTDAT}$), in modo tale che l'abilitazione di uno escluda quella dell'altro.

8. Impiego del microcomputer

Nella figura 21 è mostrato lo schema a blocchi completo del microcomputer, dove il blocco indicato come CPU Z 80 contiene anche tutti i dispositivi di buffer necessari per aumentare il pilotaggio della corrente. Infatti il fan-out della CPU, rispetto alla normale famiglia TTL, è pari solamente ad uno e ciò non è sufficiente per un dispositivo eminentemente sperimentale come è quello progettato.

Le fasi principali riguardanti l'utilizzo del microcomputer sono essenzialmente tre:
- accensione;
- memorizzazione di un programma;
- esecuzione di un programma.

Accensione

All'atto dell'accensione, che avviene connettendo al sistema la alimentazione, può essere previsto un metodo di reset automatico oppure manuale.

Memorizzazione di un programma.

Dopo aver premuto il tasto RESET, si porta il deviatore MEM/RUN nella posizione MEM. Ciò pone in diretta comunicazione la consolle con la memoria. Si imposta quindi sui sedici deviatori di indirizzo la locazione dalla quale si intende far partire il programma, e sugli otto deviatori dei dati il valore binario del codice operativo dell'istruzione o del dato. Quindi si preme il tasto ST. In questo modo si è operata una scrittura in memoria e contemporaneamente sui display compariranno i valori impostati.

Ripetendo in modo simile le precedenti fasi si possono memorizzare tutti i successivi passi del programma.

Esecuzione del programma

L'inizio dell'esecuzione di un programma può avvenire in due modi: o mediante l'uso del tasto reset oppure mediante una interruzione. Nel primo modo, poiché con il comando di reset nel μP Z 80 si azzera il PC, basta scrivere nella locazione di memoria ad indirizzo zero, **la prima istruzione del programma** che si desidera eseguire. Naturalmente, il reset manuale deve essere inviato dopo aver predisposto il deviatore MEM/RUN nella posizione RUN in modo da ridare alla CPU il controllo del bus ($\overline{BUSRQ}$ = 1).

Nel modo mediante l'interruzione, è sufficiente impostare, sugli otto deviatori dei dati, il codice operativo della RESTART che permette di indirizzare la locazione di partenza del programma.

Infatti per iniziare l'esecuzione si utilizza un impulso per l'interruzione in modo 0 della CPU che, in questo caso, attende un codice operativo ad

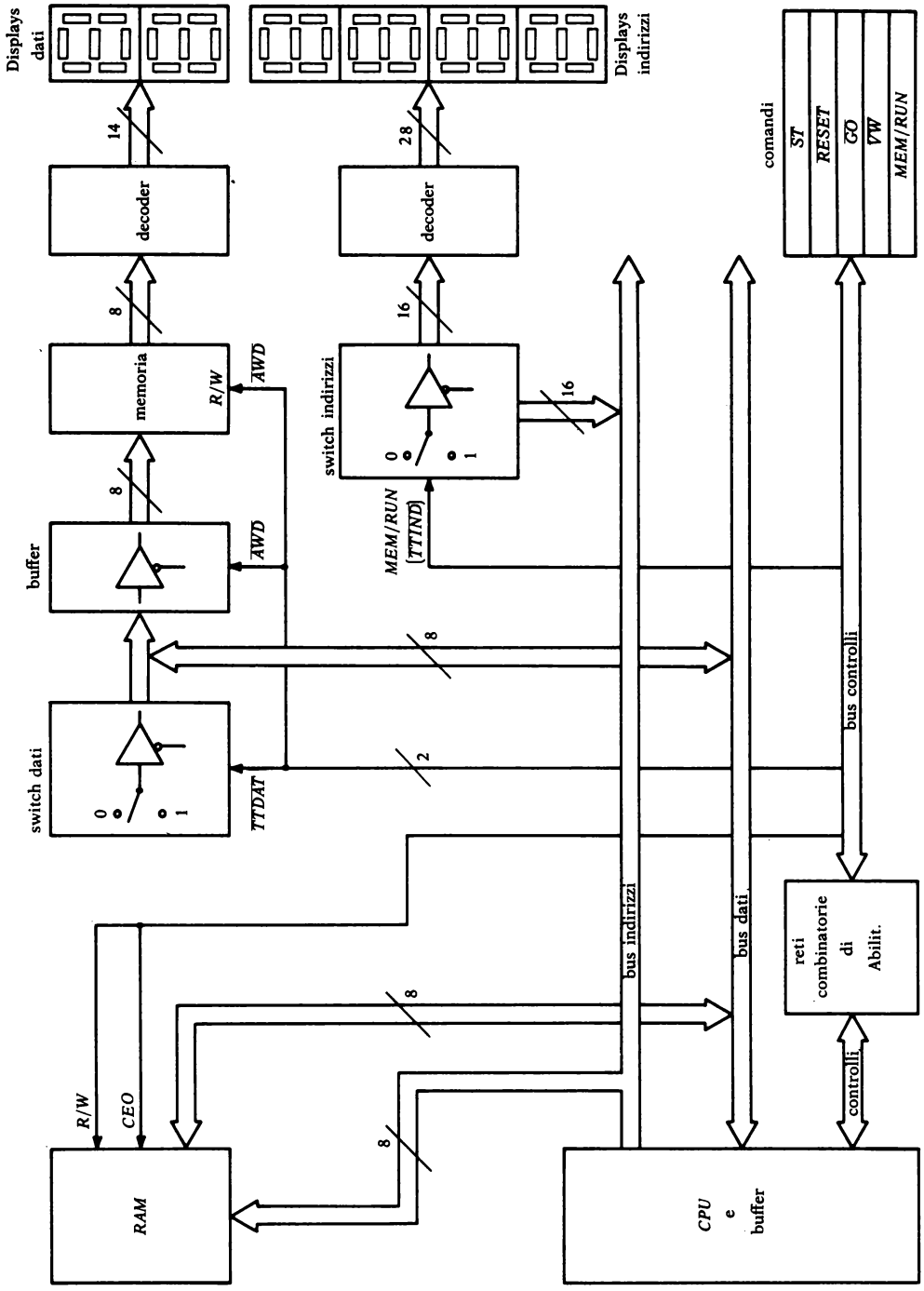


FIG. 21 - Schema a blocchi del μc.

un byte. Utilizzando le *restart*, i valori fissi delle locazioni di inizio del programma sono:

$$00_H, 08_H, 10_H, 18_H, 20_H, 28_H, 30_H, 38_H.$$

La rete che permette tale funzione è mostrata in figura 22.

Il suo funzionamento è il seguente: si imposta il codice operativo della *restart* interessata sul bus dei dati e quindi si preme il tasto GO. Ciò attiva il f-f di tipo D, che, inviando un impulso sul pin INT della CPU, provoca l'interruzione desiderata per iniziare il programma.

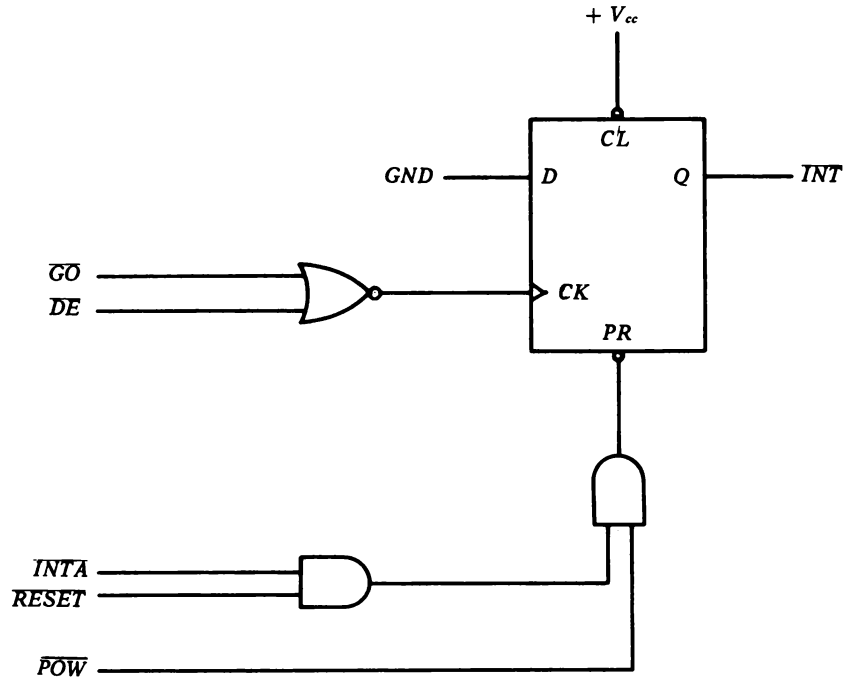


FIG. 22. - Rete per l'interruzione ed inizio esecuzione del programma.

L'impulso INT rimane basso, fino a che rimane attivo l'impulso di riconoscimento di interruzione $\overline{\text{INTA}}$ ottenuto dai segnali $\overline{\text{M1}}$ ed $\overline{\text{IORQ}}$. In definitiva, durante il periodo in cui $\overline{\text{INTA}}$ è attivo la CPU legge i dati dal bus, mentre appena torna al livello alto provvede a sconnettere il collegamento tastiera-bus dati-CPU in modo da evitare l'invio di successive interruzioni che potrebbero cancellare il contenuto del programma stesso. Infatti ad ogni interruzione il registro PC (program counter) viene salvato nello stack che raggiungerebbe dimensioni elevatissime se continuassero a sopraggiungere richieste di interruzione.

Una volta che si è certi del corretto funzionamento del sistema, si può

procedere alla scrittura di un semplice sistema operativo che semplifichi le operazioni, almeno prevedendo l'uso di una tastiera esadecimale.

All'atto della realizzazione del cablaggio è inoltre necessario mettere in atto tutti quegli accorgimenti già descritti nel quinto capitolo, e riguardanti gli ingressi inutilizzati, i disaccoppiamenti, il rumore sulle linee ed il rimbalzo dei pulsanti.

9. Il sistema operativo o monitor

Il sistema operativo è un programma che nei microcomputer commerciali risiede già nella memoria ROM o PROM, ed ha una dimensione variabile da circa 500 byte a circa 2Kbyte.

Il suo scopo principale è quello di dare un minimo di elasticità nell'impiego del μC. Le funzioni basilari consistono nel:
- controllare il terminale utilizzato per il caricamento del programma;
- eseguire il programma e caricare il PC;
- visualizzare il contenuto della memoria e dei registri;
- consentire l'arresto dell'esecuzione del programma;
- caricare su memorie di massa esterne il contenuto della RAM.

La maggior parte dei monitor è stata scritta per funzionare con un terminale tipo telescrivente, ma nei sistemi più piccoli è previsto l'uso di tastiere e di display.

Viene eseguito automaticamente all'atto dell'accensione del sistema e, per eseguire i compiti precedentemente descritti, dovrebbe essere composto dai seguenti blocchi mostrati nella figura 23:

- una routine per l'inizializzazione del sistema;
- una routine per l'interfaccia con la consolle;
- una routine per decodificare i comandi impartiti dalla consolle;
- una serie di tante subroutines quanti sono i vari comandi, per la loro esecuzione.

I tipici comandi che un monitor dovrebbe riconoscere ed eseguire sono ad esempio:
STORE - carica un dato in memoria;
LOAD - carica un dato da un lettore seriale;
OUT - carica il programma su una memoria esterna;
GO - inizia l'esecuzione del programma utente;
ESC - ritorna il controllo al monitor;
DISPLAY - mostra il contenuto della memoria;
REG - mostra il contenuto dei registri.

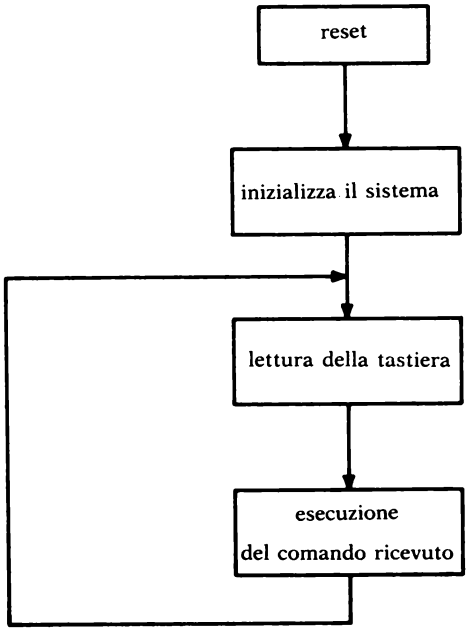


FIG. 23.-
Schema a blocchi di un semplice monitor.

Oltre a questi, che possono considerarsi fondamentali, esiste tutta una serie di subroutine che possono contribuire a trasformare un monitor scadente in uno di buona qualità.

Queste riguardano ad esempio il trattamento degli errori oppure degli operandi per quei comandi composti da più di un byte.

Per semplificare l'uso dei comandi, molto spesso la routine di lettura della tastiera è in grado di individuarli anche solamente dal primo carattere per cui tale routine potrebbe ad esempio essere configurata nel modo mostrato dalla figura 24 mentre la sua codifica risulterebbe:

```
        LD HL, RIT      carica l'indirizzo di ritorno in HL
        PUSH HL         salva nello stack HL
INIZ    CALL CONS       chiama la routine di lettura della consolle
                        che deve far parte del monitor
        CP S            confronta l'operatore letto e caricato in ac-
                        cumulatore con la lettera S
        JP Z, STORE     salta ad eseguire la routine STORE se la
                        condizione si è verificata
        CP L,           come CP S
        JP Z, L         salta alla routine LOAD se il contenuto di
                        A è uguale ad L
        .............   .....................................
        LD A, ?         carica il codice di ? in accumulatore
        CALL DIS        chiama la routine di display del contenuto
                        di A
        JP INIZ         torna all'inizio
```

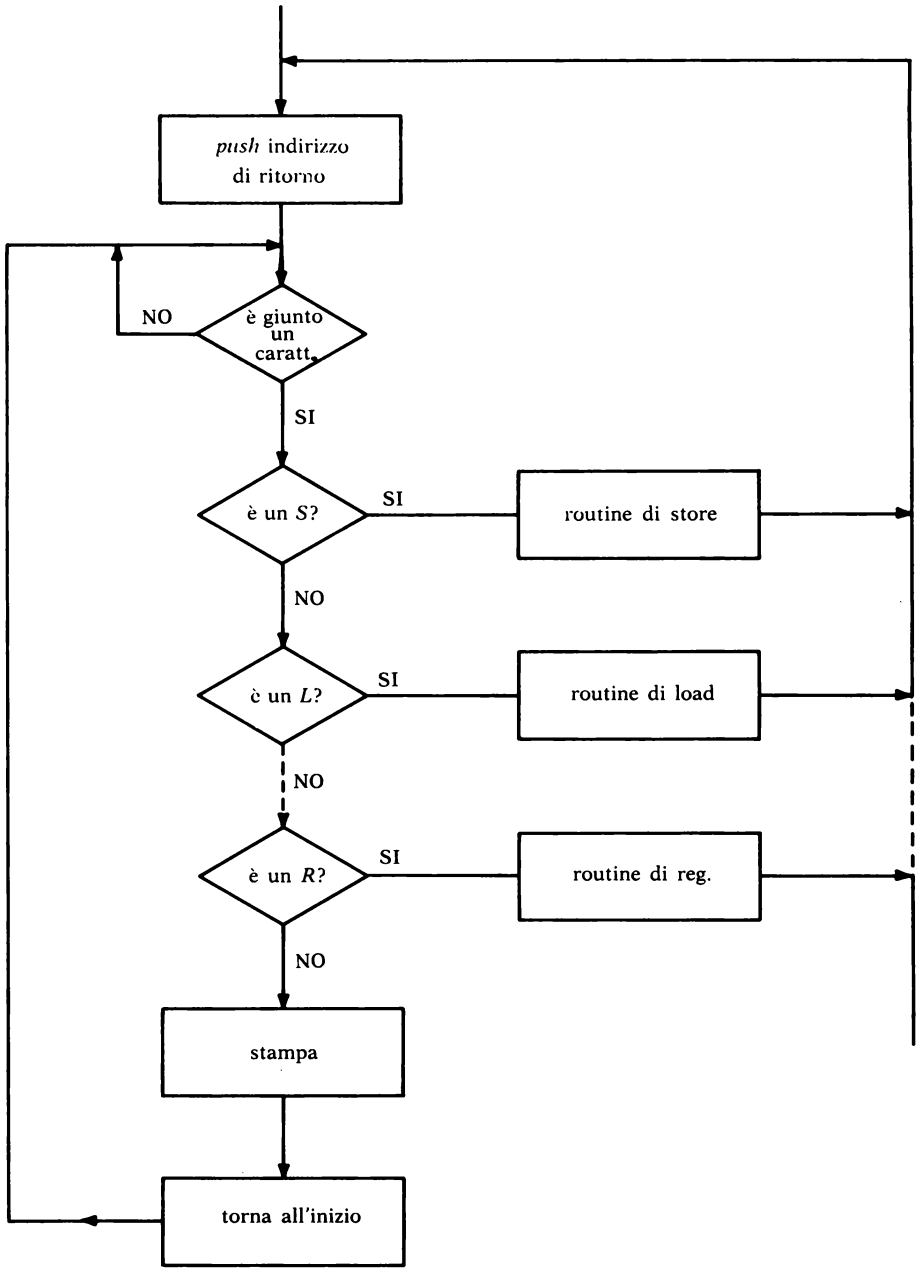


FIG. 24. - Esempio di routine per il riconoscimento e l'esecuzione dei comandi.

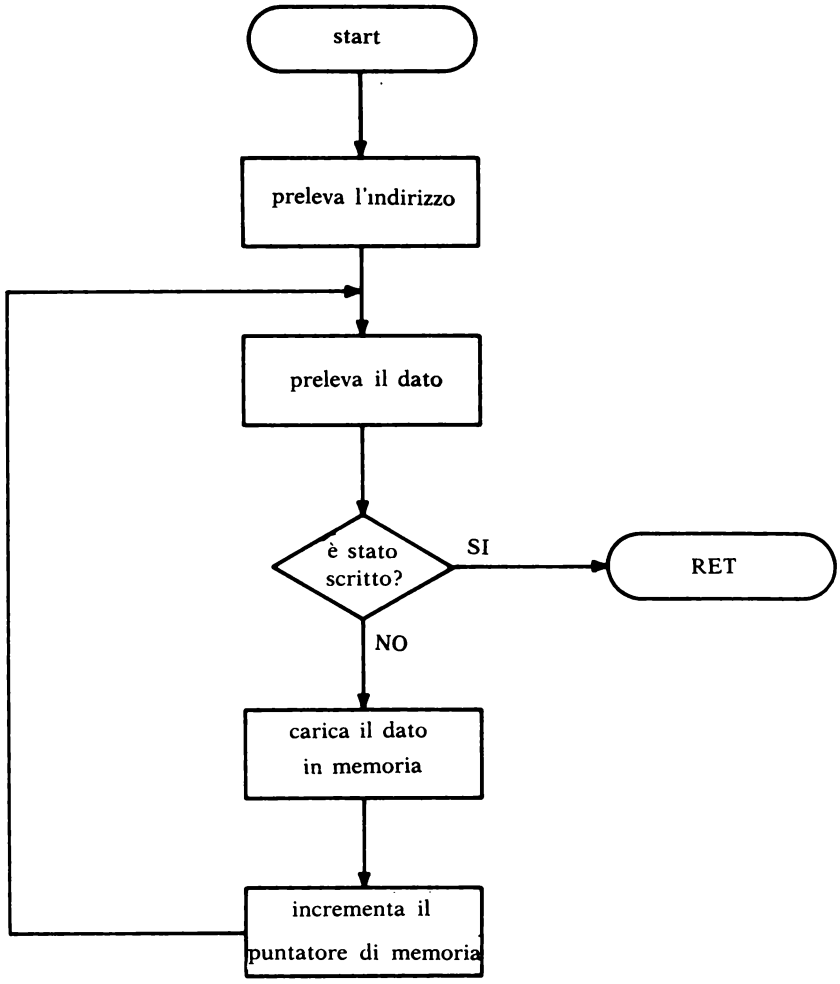


FIG. 25. - Esempio di schema a blocchi per la routine STORE.

Se nel corso di tutte le comparazioni la precedente routine rileva, ad esempio, che è stato premuto il comando S, dà inizio alla esecuzione della routine STORE che esegue il caricamento di un dato in memoria. Il suo schema a blocchi potrebbe essere quello di figura 25 o qualsiasi altra variante ritenuta più utile.

La codifica può risultare la seguente:

```
     CALL ADDR          carica in HL l'indirizzo
P1   CALL BYTE          leggi il dato dalla tastiera
     CALL COMPARE       confronta il dato con il contenuto
                        di HL
     JP NZ, P1          salta a P1 se il dato non è stato scritto
     CALL PUTDATA       scrivi il dato in memoria
     INC HL             incrementa il puntatore
     RET                torna al main program.
```

Procedendo nel modo indicato, si può giungere alla completa definizione di un intero pacchetto software costituente un esempio di sistema operativo per microcomputer.

CAPITOLO OTTAVO

MEMORIE DI MASSA

I programmi e le informazioni che non vengono usate correntemente da un sistema a microprocessore vengono memorizzati in dispositivi di memoria più lenti ma a più basso costo conosciuti con il nome di *memorie di massa*.

L'evoluzione dei sistemi di memoria di massa ha come inizio i primi anni del '60 quando per memorizzare grandi quantità di dati, i mezzi disponibili erano le schede perforate, il nastro perforato ed i nastri magnetici. Ciò implicava che la memorizzazione ed il recupero dei dati fosse particolarmente lento se paragonato agli standards moderni. Attualmente la panoramica dei mezzi di supporto per la memorizzazione dati utente per quanto riguarda sistemi a microprocessore comprende le memorie a scorrimento di carica a semiconduttore o CCD e le memorie di massa magnetiche (floppy disk, bolle magnetiche, cassette magnetiche).

Le memorie di tipo magnetico offrono rispetto a quelle a semiconduttore prestazioni più limitate, e per poter essere competitive debbono quindi avere un costo piuttosto contenuto.

In fig. 1 è riportato un diagramma in cui vengono messi a confronto il tempo di accesso ed il prezzo di vari tipi di memorie. Si può notare dall'esame della figura come le memorie CCD e quelle a bolle magnetiche stiano colmando un vuoto tra le memorie ultraveloci, di tipo bipolare e ad accesso causale, e quelle molto più lente di tipo dinamico.

1. Memorie a scorrimento di carica o CCD (Charge Coupled Device)

Le memorie a semiconduttore finora esaminate (vol. Elettronica Digitale) hanno la caratteristica comune di essere ad accesso casuale (random), cioè il tempo necessario per una operazione di lettura o scrittura risulta indipendente dalla posizione fisica, entro la matrice di memoria, della cella a cui si deve accedere. Notevoli risparmi si possono ottenere nella progettazione

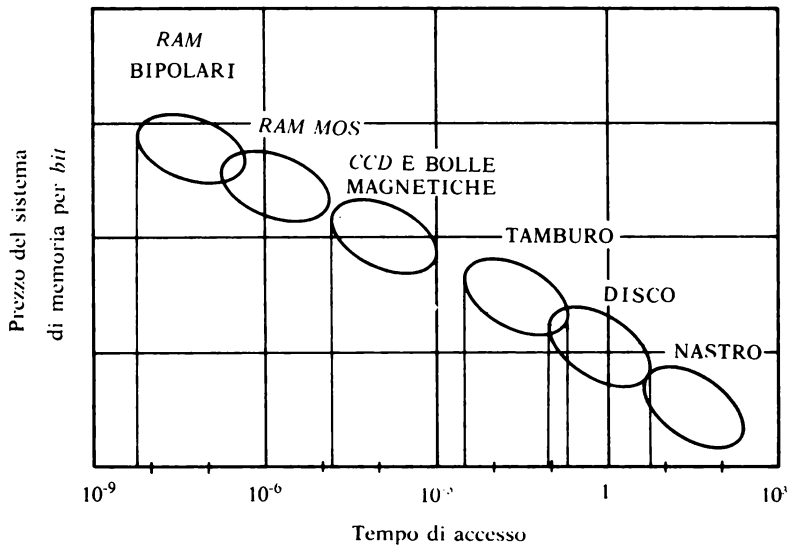


FIG. 1 - Tempo di accesso di vari tipi di memorie.

della memoria se non è strettamente necessario l'accesso casuale. Per esempio una memoria usata per rigenerare le informazioni presentate in un terminale video standard, che è esplorato punto per punto in un disegno lineare ripetitivo, non ha bisogno di una memoria ad accesso causale.

Le memorie CCD (realizzate con tecnologia MOS) sono memorie ad accesso sequenziale in cui i bit memorizzati circolano come se si trovassero in una conduttura chiusa.

Ogni bit memorizzato viene trasferito in sequenza attraverso 64 o più locazioni di memoria tra il tempo in cui viene scritto nella memoria ed il tempo iniziale in cui esso diventa disponibile per la lettura. La velocità con la quale i bit sono trasferiti da una locazione all'altra in una memoria CCD è circa uguale al tempo di ciclo di una RAM, cosicché il tempo di accesso più lungo per una memoria seriale con 128 locazioni è uguale a 128 volte il tempo di ciclo di una memoria random.

Ne risulta che i tempi di accesso per le CCD sono sensibilmente alti se raffrontati con quelli delle memorie RAM ad uguale capacità di memorizzazione, ma sempre inferiori a quelli di altre memorie di massa quali, per esempio, i floppy-disk o le cassette magnetiche.

Il costo per bit di una memoria CCD risulta di gran lunga inferiore a quello di una RAM dinamica e rispetto a quest'ultima l'area totale di silicio per bit per i componenti di una memoria completa è due o tre volte più piccola.

Attualmente sono reperibili memorie CCD con capacità di immagazzinamento di 64 K in un chip di 17,5 mm^2 con consumo di qualche μW per bit.

L'elemento base di un dispositivo CCD è costituito dall'insieme di tre piccoli condensatori realizzati a tecnologia NMOS, il principio di funzionamento è il seguente:
l'acquisizione dell'informazione viene immagazzinata sotto la forma di carica positiva nel primo condensatore (fig. 2a) nella regione al di sotto del relativo elettrodo a cui è applicata una tensione negativa V_1 mentre agli elettrodi 2 e 3 viene applicata una tensione negativa V_2 (con $|V_1| < |V_2|$). Si viene così a creare una regione di svuotamento (indicata dal tratteggio) ed una buca di potenziale in grado di trattenere le cariche minoritarie, iniettate e regolate nel canale del dispositivo per mezzo di una tensione esterna. La presenza o l'assenza di queste cariche rappresenta l'informazione binaria.

Variando la tensione agli elettrodi, mediante impulsi di clock su tre fasi (4 fasi per dispositivi CCD della Intel), si può far scorrere la carica immagazzinata sotto il secondo elettrodo (fig. 2b) e quindi cristallizzarla in tale posizione (fig. 2c). A questo punto la carica immagazzinata si è spostata di un elettrodo verso destra.

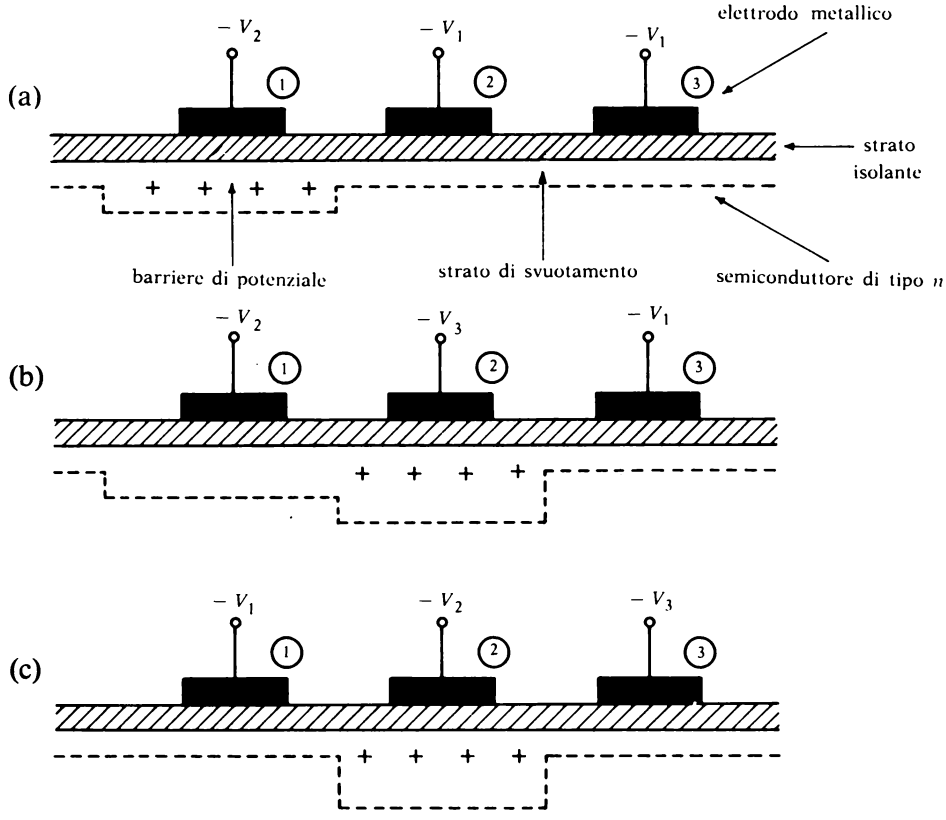


FIG. 2. - Principio di funzionamento di una memoria CCD.

Una limitazione di un dispositivo CCD è la dispersione delle cariche nel substrato durante lo scorrimento da un elettrodo al successivo. La percentuale di carica trasferita ad ogni operazione di scorrimento raggiunge circa il 99,9%.

Per effetto di questa dispersione i dispositivi CCD necessitano di periodiche operazioni di lettura e rinfresco, operazioni che vengono effettuate mediante circuiti di rigenerazione ed amplificazione. In fig. 3 è riportato lo schema funzionale della memoria a scorrimento di carica TMS 3064 della Texas.

functional block diagram

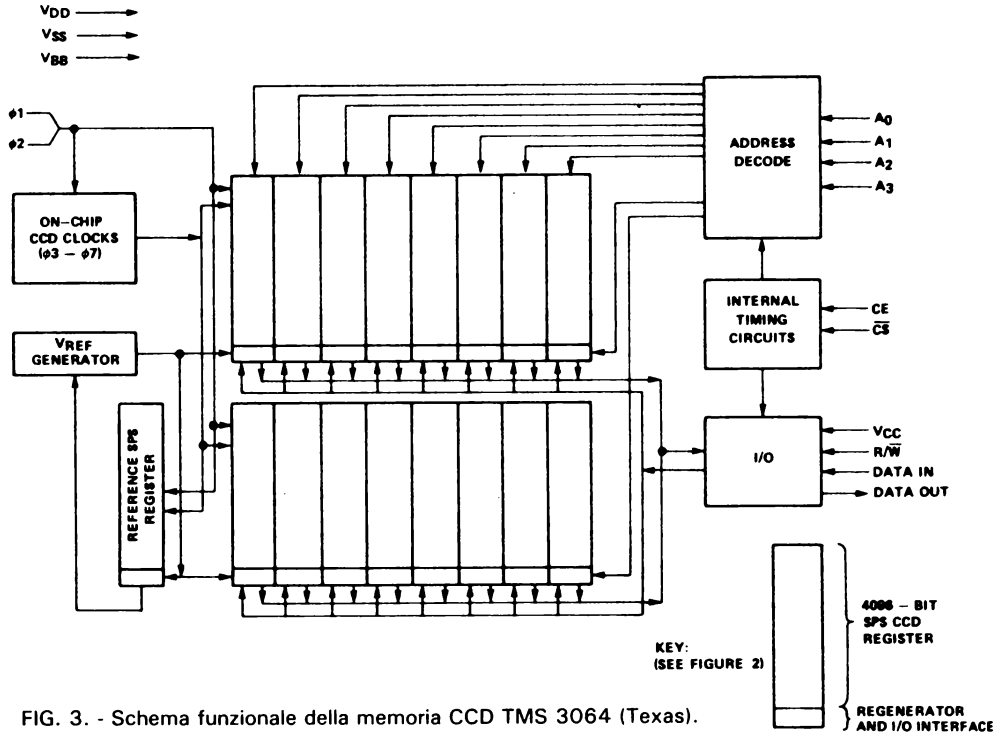


FIG. 3. - Schema funzionale della memoria CCD TMS 3064 (Texas).

La TMS 3064 è organizzata internamente in 16 anelli indirizzabili ognuno costituito da 4096 bit. Le caratteristiche principali di questa memoria sono:

- bassa potenza dissipata (alla massima frequenza circa 280 mW)
- clock a due fasi
- direttamente compatibile con la TTL su tutti gli ingressi ad eccezione di quelli di clock Φ_1, Φ_2 e di quello di chip-enable (CE).

- gamma di lavoro da 1 MH_z a 5 MH_z
- tempo di ciclo di lettura o scrittura uguale a 200 ns.

Ogni anello della memoria è costituito da un registro a scorrimento CCD seriale-parallelo-seriale (SPS), da un circuito rigeneratore per il rinfresco dei dati e da un circuito di interfaccia per la logica esterna (fig. 4).

Un ulteriore registro non indirizzabile, il diciassettesimo, è poi usato insieme ad un generatore di tensione di riferimento per rinfrescare la quantità di carica ad ogni anello. Sebbene per il funzionamento della memoria siano richiesti sette differenti clock, soltanto due di questi (Φ_1 e Φ_2) debbono essere forniti dall'esterno in quanto gli altri cinque sono generati all'interno dello stesso chip. La funzione dei clock Φ_1 e Φ_2 è quella di rinfrescare e di far circolare i dati immagazzinati negli anelli.

All'interno di ogni anello il dato è rilevato serialmente per mezzo dei clock Φ_1 e Φ_2 che forniscono impulsi fin tanto che il bit desiderato non sia stato shiftato all'uscita del registro SPS selezionato. All'uscita del registro il dato è rinfrescato da un rigeneratore e poi rinviato all'ingresso del registro stesso. Il dato presente all'uscita di ciascun rigeneratore è letto selezionando l'apposito indirizzo e portando i pin R/$\overline{W}$ e CE alti con il chip-select (CS) al livello logico basso.

Il dato in uscita sarà poi valido una volta trascorso il tempo di ritardo dovuto al rise-time del chip-enable.

La scrittura di un dato in un particolare anello è effettuata portando agli ingressi CS e R/$\overline{W}$ (read/write select) al livello logico basso con CE al livello logico alto. Un determinato bit può essere letto e quindi modificato nello stesso ciclo di clock eseguendo una operazione detta di modifica-lettura-scrittura (RMW). L'RMW consiste in una prima operazione di lettura come quella appena vista e quindi nell'invio del dato in ingresso con R/$\overline{W}$ basso mentre il chip-enable è ancora al livello logico alto. Il ciclo RMW può essere usato per interscambiare i contenuti di due memorie TMS 3064 connettendo l'uscita dati dell'una all'ingresso dati dell'altra o viceversa.

Struttura interna di un SPS CCD

Come si è visto precedentemente, ogni blocco indirizzabile della TMS 3064 contiene 4096 bit di memoria organizzata come registri a scorrimento SPS.

Il dato immagazzinato è shiftato nell'SPS lungo un registro d'ingresso seriale a 32 bit, è quindi demultiplexato in 32 registri a scorrimento paralleli ognuno di 127 bit, e finalmente multiplexato in un registro seriale d'uscita a 32 bit che shifta il dato al rigeneratore. Il rigeneratore rileva la piccola quantità di cariche che devono essere trasferite e converte i livelli di queste cariche in livelli di tensione che sono confrontati con una tensione di riferimento (V_{REF}). L'uscita del rigeneratore invia, poi, durante un ciclo di lettura o di ricircolazione, all'ingresso dell'SPS un segnale digitale

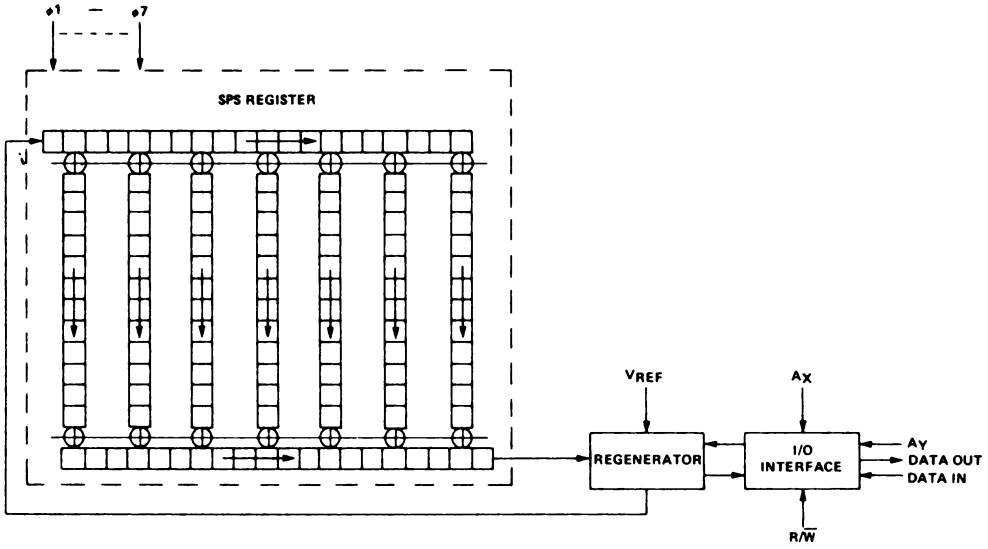


FIG. 4. - Struttura di un SPS della CCD TMS 3064

per il rinfresco del dato memorizzato. Il trasferimento del dato da e per l'anello è controllato da un circuito di interfaccia ingresso-uscita (I/O).

2. Memorie a bolle magnetiche.

Le memorie a bolle sono memorie ad accesso seriale, allo stato solido, che sfruttano la mobilità delle bolle magnetiche, ovvero domini microscopici di polarizzazione magnetica in un sottile film magnetico di ortoferrite o granato. In presenza di un campo magnetico stazionario di adeguata intensità, con le linee di forza perpendicolari al piano del film, i domini di polarizzazione (per esempio verso l'alto) sono stabili entro un'area maggiore di polarizzazione opposta (verso il basso). Infatti l'applicazione di un campo magnetico (fig. 5) provoca il restringimento dei domini di polarità opposta al campo e la dilatazione dei domini con la stessa polarità. Si ottengono delle bolle magnetiche quando i domini si sono ridotti a piccoli cilindri con diametro dell'ordine dei micron.

I domini si possono muovere sul piano del film applicando campi magnetici più deboli ad angolo retto rispetto al campo principale. Quando i domini sono organizzati in modo appropriato, possono costituire una memoria a bolle magnetiche. Se un dominio con polarizzazione verso l'alto viene usato per designare un 1 logico, l'assenza di tale dominio può essere considerato come uno 0 logico. I domini si possono formare od eliminare con un anello elettromagnetico a spira unica sulla superficie del film ma-

gnetico: per distruggere una bolla basta aumentare localmente il campo di polarizzazione fino a raggiungere un determinato valore caratteristico della memoria a bolle considerata; una bolla è invece generata mediante l'invio di un impulso su un'opportuna spira.

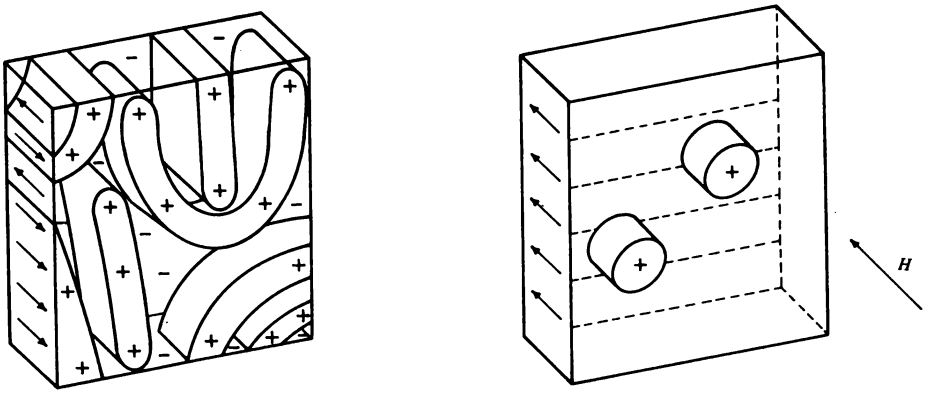


FIG. 5. - Formazione di bolle magnetiche.

Un modo molto pratico per realizzare una memoria a bolle è quello di organizzarla come un registro a scorrimento, facendo viaggiare le bolle in modo sincrono lungo percorsi chiusi prestabiliti. Il sistema più semplice per raggiungere questo scopo è quello di tracciare un percorso per mezzo di piastrine di permalloy opportunamente sagomate, per esempio a forma di T ed I (fig. 6).

Per far muovere le bolle si applica un campo magnetico rotante nel piano del substrato. Le piastrine si polarizzano parallelamente al campo magnetico, attirando via via le bolle lungo il percorso voluto. Ad una rotazione completa del campo magnetico corrisponde l'avanzamento di un passo da parte delle bolle, bolle che debbono essere opportunamente distanziate (all'incirca di 4 ÷ 5 diametri) affinché non ci sia interazione fra di loro.

Per superare questa limitazione è stata proposta una nuova soluzione, in cui le bolle sono compresse l'una accanto all'altra secondo una struttura periodica. Non essendoci spazi vuoti, l'informazione non può più essere rappresentata mediante la presenza o l'assenza di una bolla, ma in questo caso occorre far uso di due diversi tipi di bolle associati ai due valori binari.

Si parla così di «bubble-lattice-storage». Una caratteristica molto importante delle memorie a bolle consiste, a differenza di quanto avviene per le memorie CCD che sono di tipo volatile, nel fatto che l'informazione memorizzata viene mantenuta anche quando viene tolta l'alimentazione esterna. La polarizzazione delle bolle è mantenuta per mezzo di magnete

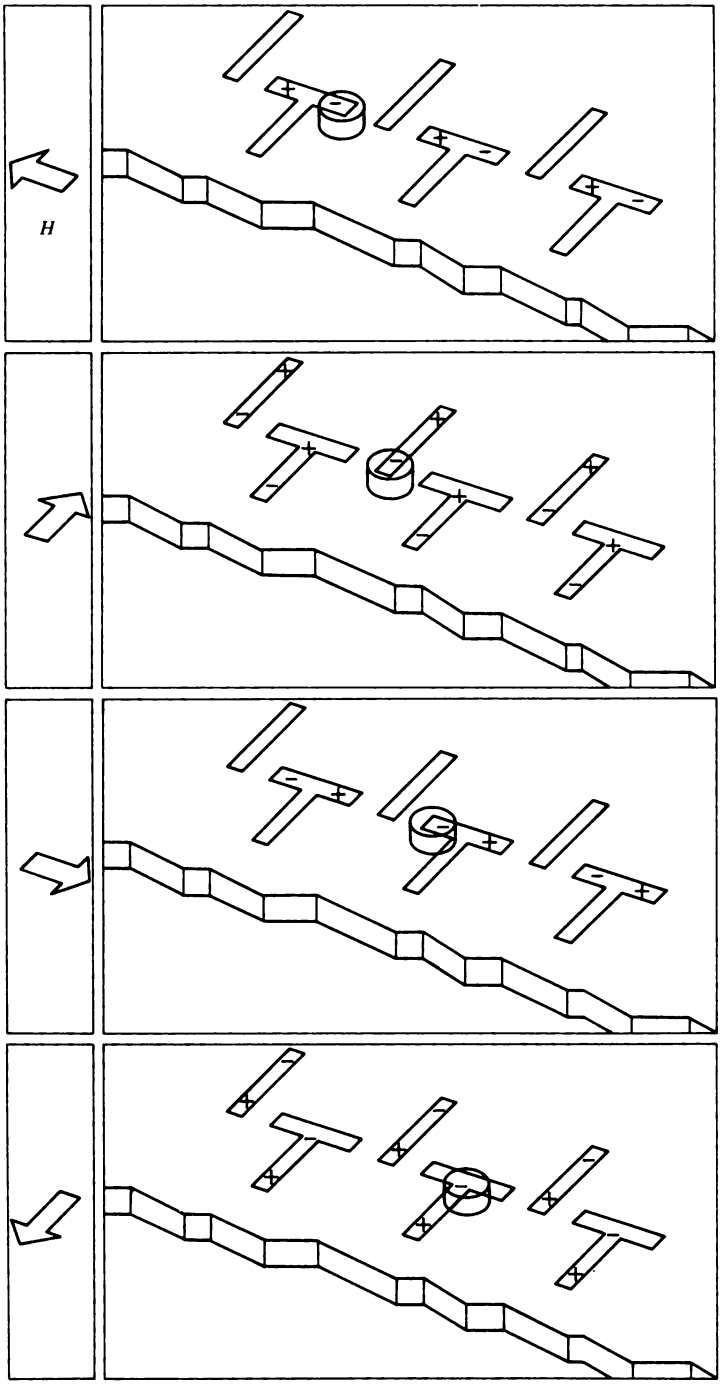


FIG. 6. - Spostamento della bolla lungo il substrato per effetto di un campo magnetico rotante.

permanente che conserva il campo magnetico stazionario. I circuiti elettronici sono necessari per guidare la formazione delle bolle, azionare gli anelli di pilotaggio e amplificare il segnale di uscita che si ottiene per induzione elettromagnetica o per effetto magnetoresistivo quando una bolla passa sotto un sensore. Questi circuiti sono alimentati durante le operazioni di lettura/scrittura. Essendo organizzate serialmente le memorie a bolle hanno un tempo di accesso che varia a seconda delle posizioni di memoria e dalla velocità massima di spostamento. Nelle attuali memorie i cammini seriali hanno una lunghezza che varia da circa 10 a 1000 o più posizioni di memoria e le velocità di spostamento variano da una frazione di μs a diversi μs.

Per aumentare la capacità di memoria senza dover aumentare proporzionalmente il tempo di accesso le memorie a bolle sono strutturate in anelli secondari (minor loops) collegati ad un anello principale (major loop). In fig. 7 è mostrato il diagramma relativo ad una memoria a bolle magnetiche, organizzata in questo modo, di 92304 bit che illustra il modo di circolazione delle bolle.

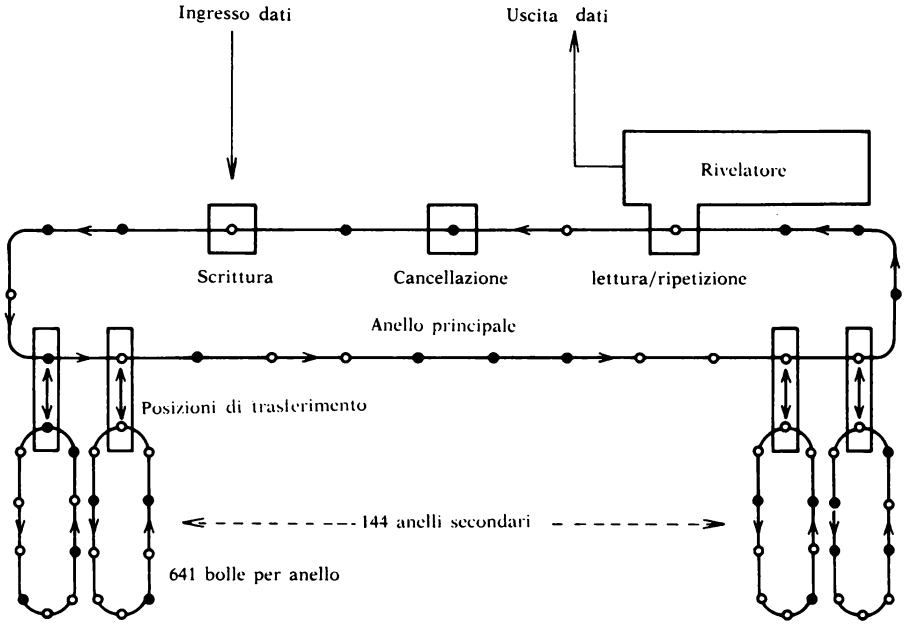


FIG. 7. - Circolazione di bolle in una memoria di 92304 bit

L'anello principale contiene un singolo blocco di dati, che consiste di 1 e 0 (bolla o nessuna bolla) che si stanno, leggendo, ripetendo o cancellando. In questo particolare dispositivo il blocco dati contiene 144 bit. Nel ciclo di lettura i 144 bit prima entrano nell'anello principale, dove vengono

trasferiti simultaneamente, ad un dato segnale, nei 144 anelli secondari, un bit per ogni anello. Ogni minor loop può contenere a sua volta 641 bolle. La capacità totale del dispositivo è quindi di 144 × 641 = 92304 bit. Nel ciclo di lettura i 144 bit sono trasferiti, per mezzo di un opportuno segnale, dagli anelli secondari nell'anello principale che li porta sotto la testina di lettura.

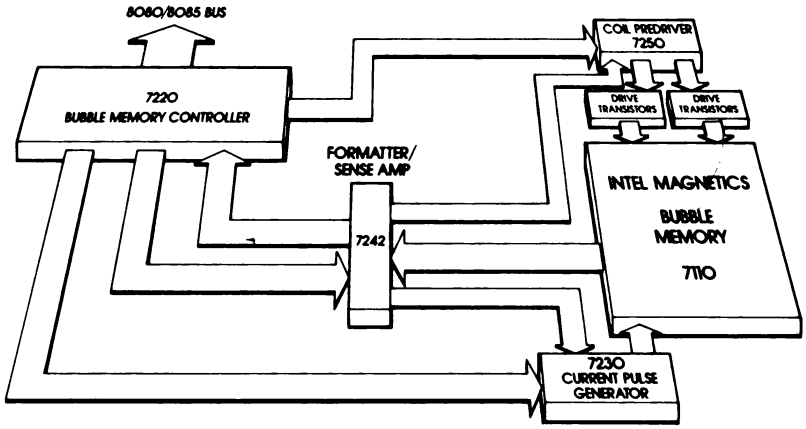


FIG. 8. - Schema a blocchi di un sistema a bolle magnetiche di 128 K bytes compatibile con un sistema a microprocessore.

In fig. 8 è infine riportato lo schema a blocchi di un sistema singolo di memoria a bolle da 1 megabit (1048576 bit), della Intel, direttamente interfacciabile al bus dati di un microprocessore ad otto bit mediante componenti specifici, che fanno da supporto alla memoria a bolle 7110 strutturata in 256 anelli secondari ognuno dei quali contenente 4096 bit.

3. Nastri Magnetici

Attualmente i nastri magnetici vengono impiegati come memorie sequenziali di massa nei sistemi a microprocessore nei quali il costo è un fattore prioritario rispetto alla velocità dell'apparato.

Le cassette magnetiche utilizzate per immagazzinare le informazioni in un sistema a microcomputer sono del tutto simili a quelle usate negli altri campi, è sufficiente soltanto che siano di ottima qualità. Naturalmente anche il registratore impiegato deve essere di ottima fattura, ciò per minimizzare gli errori di lettura/scrittura. Resta inteso inoltre che occorre un apposito circuito di interfaccia tra il sistema ed il registratore a cassette. Le informazioni vengono memorizzate sul nastro come suono associando

a due diversi toni i valori di 0 ed 1. Fino a qualche tempo fa le informazioni invece venivano immagazzinate in forma digitale come sequenza di punti magnetizzati; tale metodo però è stato abbandonato in quanto sensibilmente più costoso rispetto al primo poiché richiedeva l'uso di un registratore particolare. Le informazioni sono immagazzinate nel nastro magnetico come sequenza di records (messaggi), la lunghezza del record può essere variabile e al limite coincidere con la lunghezza di tutto il nastro magnetico.

Un record logico può essere memorizzato come un unico record fisico sulla cassetta, oppure può essere suddiviso in più record fisici. Tra un record e l'altro deve esserci una zona di separazione (gap) in cui non debbono essere immagazzinate informazioni utili; ne segue quindi che aumentando il numero dei record diminuisce la quantità di informazione utile memorizzabile sulla cassetta magnetica. La memorizzazione di record molto corti, però, se da una parte limita la capacità di memorizzazione della cassetta magnetica, dall'altra rende molto più rapida la scoperta di eventuali errori di registrazione.

Il tempo di accesso in una cassetta magnetica dipende ovviamente dalla lunghezza del nastro e può essere anche di una decina di minuti (una decina di secondi per le minicassette) e comunque risulta piuttosto elevato.

Per ridurre il tempo e migliorare la velocità di registrazione si possono utilizzare registratori particolari in cui è stato eliminato il sistema di far avviare o fermare il nastro magnetico nelle operazioni di lettura/scrittura. Il modello Thorn Emt, per esempio, riesce a registrare fino a 61 M byte mediante una rotazione continua del nastro magnetico a una velocità di 50 ÷ 100 pollici/secondo, dopo aver eliminato i rulli di trascinamento piuttosto costosi.

4. Floppy Disk

Il floppy disk o dischetto flessibile, immesso sul mercato agli inizi degli anni '70 dalla IBM, è una memoria di massa utilizzata soprattutto nei sistemi basati a microprocessore come, per esempio, i sistemi di sviluppo (cap. 9°).

Alcuni vantaggi del floppy disk rispetto ad altre memorie di massa sono:

- basso costo
- alta affidabilità e durata
- piccolo ingombro e facilità di trasporto
- compatibilità dei dati immagazzinati con altri sistemi più o meno grandi.

La capacità di memorizzazione del floppy disk (chiamato anche diskette) si aggira attualmente intorno ai 120 K byte per un floppy disk da 5,25 pollici (1 pollice = 2,54 cm) a singola densità e faccia, fino a 1,6 M byte

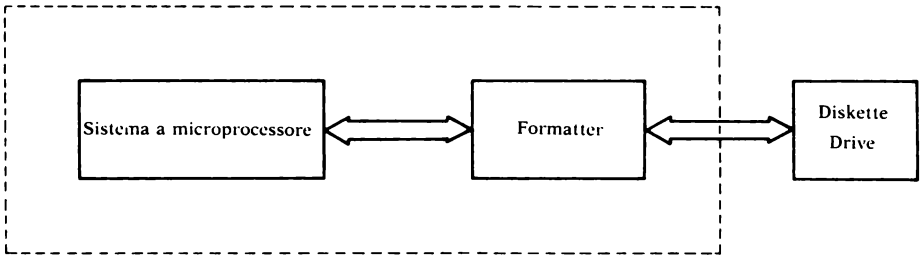


FIG. 9. - Sottosistema a floppy disk.

per un floppy disk da otto pollici a doppia densità e faccia; valori correnti per quest'ultimi sono sui 250 K byte per faccia.

In fig. 9 è mostrato lo schema a blocchi di un sottosistema floppy disk a microprocessore. Il diskette drive (o più semplicemente drive) è un periferico a basso costo che esegue le funzioni elettromeccaniche di lettura/scrittura necessarie per registrare ed estrarre le informazioni dal floppy disk.

Poiché i dati sul disco sono memorizzati serialmente è necessaria una logica di interfaccia per effettuare la conversione seriale-parallela; l'hardware che permette tale conversione è detto *formatter*. Il formatter inoltre esegue la bufferizzazione tra il sistema a microprocessore ed il disco.

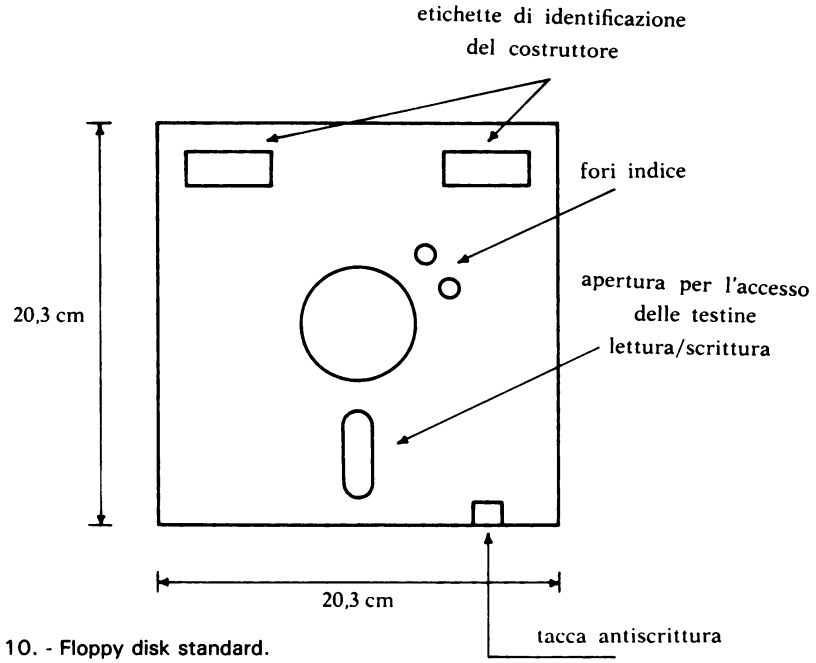


FIG. 10. - Floppy disk standard.

Il floppy disk è formato da un disco flessibile con un supporto plastico (in genere costituito da mylar), su cui è depositato un materiale magnetico, racchiuso in un apposito contenitore rigido (fig. 10). La dimensione del contenitore distingue i due tipi di floppy attualmente esistenti: se il formato è di otto pollici si parla di floppy disk *standard,* nel caso di 5,25 pollici invece il floppy disk viene chiamato comunemente *minifloppy.*

In genere la capacità di memorizzazione del minifloppy è circa un terzo di quello standard con costo sensibilmente inferiore. Per quanto riguarda il tempo medio di accesso alle informazioni questo si aggira intorno agli ottanta nanosecondi per i floppy standard e a 100 ns per i minifloppy. Le informazioni vengono immagazzinate sulla superficie del disco come sequenza di impulsi magnetici disposti lungo un insieme di tracce concentriche (77 per il tipo standard, 35 per il minifloppy) numerate.

La lettura o la scrittura delle informazioni viene effettuata per mezzo di una testina magnetica piatta capace di rilevare (in fase di lettura) o fornire (in fase di scrittura) gli impulsi magnetici. A differenza delle cassette magnetiche in cui l'accesso all'informazione è di tipo sequenziale, nel floppy disk l'accesso è di tipo random in quanto è possibile selezionare una particolare traccia mediante l'associato numero di traccia, numero che può essere indirizzato in base alla distanza della traccia dal bordo del disco.

Il disco è poi diviso in settori (di solito 26 per lo standard e 16 per i minifloppy) ognuno costituito da 128 byte, per permettere un accesso più rapido all'informazione all'interno di una traccia, cosicché l'informazione sulla superficie del disco può essere identificata per mezzo del numero di traccia e del numero di settore. Nei floppy disk chiamati *hard sectored disk* l'indirizzo di ciascun settore è determinato mediante una fotocellula che rileva fisicamente la presenza di un foro sul disco all'inizio di ognuno

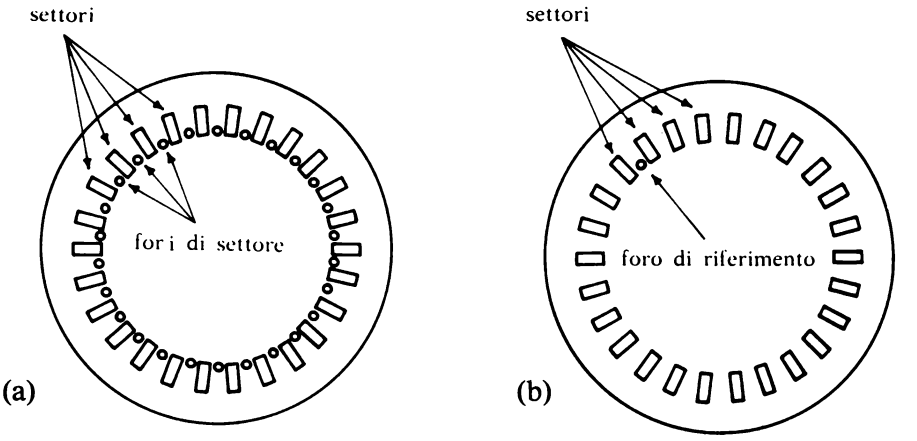


FIG. 11. - (a) hard sectored disk (b) soft sectored disk.

di essi (fig. 11a). Nei floppy disk *soft sectore disk* (fig. 11b) viene invece usato un solo foro di riferimento come inizio traccia, e l'inizio dei singoli settori è determinato per mezzo di una circuiteria esterna (generalmente integrati MSI come contatori e registri) posta normalmente nella parte di interfaccia del periferico.

Formattazione

Un floppy disk vergine ha la superficie completamente priva di informazioni, prima di poter scrivere i dati sul supporto è necessaria una operazione detta di formattazione.

La formattazione è la procedura che realizza la scrittura di un campo indirizzo e di un campo dati in ciascun settore di ogni traccia del disco. In particolare il processo di formattazione permette all'utente di verificare direttamente la qualità del supporto magnetico.

Codifica dell'informazione

Come si è visto precedentemente le informazioni vengono memorizzate, nei settori in cui è diviso il disco, serialmente come stringa di bit. Ogni bit dell'informazione però prima di essere registrato sul disco deve essere opportunamente codificato a seconda del tipo di supporto magnetico e della banda del canale di lettura. Lo spazio fisico del disco riservato al bit d'informazione viene detto comunemente cella. Nelle celle possono trovarsi sia bit di informazione vera e propria sia impulsi di clock, questi ultimi necessari per la sincronizzazione del bit d'informazione.

Tra i diversi tipi di codifica oggi esistenti i più usati sono: l'FM (Modulazione di frequenza) per drives a singola densità e l'MFM (Modulazione di frequenza modificata) per drives a doppia densità di memorizzazione.

Nella FM il bit di clock è sempre scritto all'inizio di ogni cella e il bit di informazione al centro della cella (fig. 12a). La cella è lunga 4 μs (per un floppy standard) e quindi il bit di informazione si trova esattamente 2 μs dopo il bit di clock.

La presenza in ogni cella del bit di clock semplifica fortemente la codifica e la successiva decodifica delle informazioni in quanto per separare i dati basta generare, mediante un apposito circuito, una finestra della durata di 2μs sincronizzata con il bit di clock che è sempre presente.

Nella MFM invece la dimensione della cella viene dimezzata (2 μs). Di conseguenza questa tecnica permette una doppia densità di memorizzazione rispetto alla FM, anche se il circuito di separazione dei dati diventa più complicato. Infatti nella tecnica modulazione di frequenza modificata mentre i bit di informazione sono ancora situati al centro della cella, i bit di clock invece sono memorizzati all'inizio della cella solamente se il bit di informazione sulla cella corrente e su quella precedente vale 0 (fig. 12b).

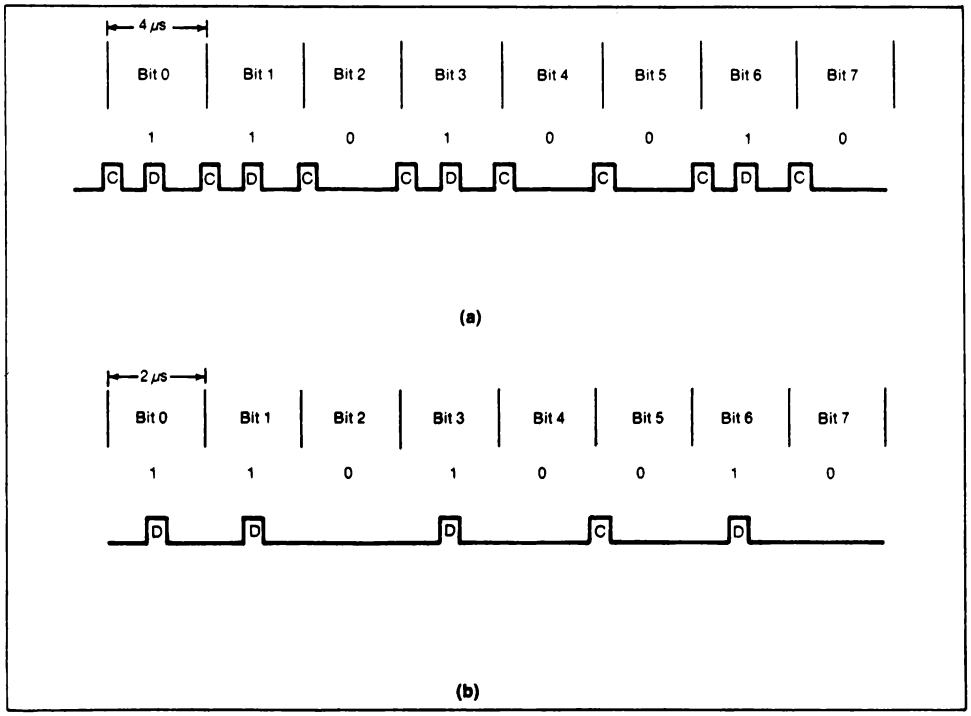


FIG. 12. - Tecnica di codifica MF (a) e MFM (b).

Nei grandi sistemi, dove è richiesta una grande capacità di memorizzazione a basso costo, vengono di solito usati dischi rigidi di tipo *Winchester,* di 14 od 8 pollici, in grado di memorizzare dalle decine alle centinaia di M byte.

La Rotating Memory System ha attualmente immesso sul mercato dischi rigidi di cinque pollici con capacità di memorizzazione fino a 18 M byte.

5. Sistemi vari di memorizzazione

Si stanno attualmente affacciando sul mercato altri tipi di memorie di massa basate su vari sistemi, tra questi vengono qui di seguito descritti molto brevemente i sistemi a laser e quelli a microfilm.

Sistemi a laser

Questi sistemi impiegano un laser per creare buche microscopiche su di

una superficie plastica, sia in forma di traccia su disco o di righe su di una scheda. Un raggio laser molto fine brucia dei buchi neri di circa 5 micron di diametro secondo un certo percorso, che viene letto dal dispositivo di lettura, in contrasto alle aree circostanti altamente riflettenti. In un disco da 5 pollici è possibile memorizzare con una velocità di 10 M byte al secondo fino a 5 M byte. L'informazione memorizzata può poi essere letta con una velocità di circa 4,3 M byte al secondo. Un grosso inconveniente delle memorie realizzate secondo questo sistema è quello che una volta scritta l'informazione quest'ultima può poi essere solamente letta, in pratica queste memorie si comportano come memorie ROM.

Sistemi a microfilm

Un tipo di memoria di massa a microfilm, per l'immagazzinamento di molte pagine di testo da recuperare e visualizzare, impiega dei video che possono mostrare pagine da una speciale memoria microfilm che può contenere fino a quattro milioni di immagini.

Un sistema di indicizzazione, attivato dal terminale e dalla tastiera, genera una ricerca dell'immagine su film da visualizzarsi: una telecamera ad alta definizione (circa 2000 linee) ne fornisce una fotografia nel tempo di dieci secondi. La fotografia viene tenuta nell'unità di visualizzazione elettronica per la lettura sullo schermo, dove può venire aggiornata, se necessario, rifotografata con lo stesso tipo di macchina e quindi il film viene ricaricato nella memoria di massa. Per i tipi di memorizzazione a lungo termine di dati i microfilm o microfiches (un foglio delle dimensioni di una cartolina che può contenere parecchie immagini tipo fotografia) devono essere tenuti seriamente in considerazione per il loro rapporto costo-rendimento rispetto ad altri sistemi di memorizzazione elettronica in forte sviluppo.

CAPITOLO NONO

LA STRUMENTAZIONE DIGITALE

1. Introduzione

Con l'avvento dei microprocessori, il lavoro di progettazione si è sempre più orientato verso il sistema che verso il singolo componente rivoluzionando di conseguenza anche i metodi di misura e di test e rendendo ancor più ampio il divario tra le tecniche dell'elettronica analogica e quella digitale.

I sistemi logici sono costituiti dall'insieme di due elementi fondamentali: l'hardware ed software. La crescita di un complesso sistema può pensarsi suddivisa in otto differenti funzioni principali, e ad esse sono associati diversi strumenti che coprono determinati settori. Nella figura 1 è mostrata l'area di applicazione di alcuni strumenti fondamentali, dal voltmetro digitale (DVM) e oscilloscopio, necessari per la creazione di un hardware di base, fino ai sistemi di sviluppo per seguirne la corretta interazione con il software.

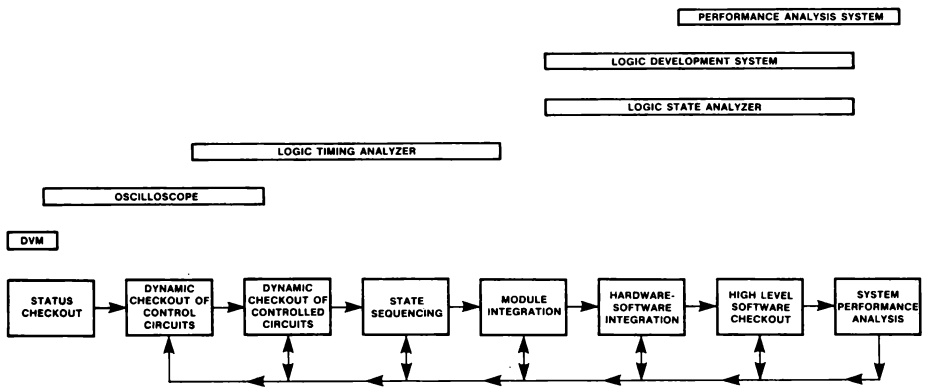


FIG. 1. - Aree coperte da alcuni strumenti elettronici.

Nei paragrafi successivi, verrà illustrato il principio di funzionamento ed il principale campo di applicazione di alcuni di essi.

2. Sonde logiche

Le sonde logiche, nelle loro molteplici diversità costruttive, costituiscono la soluzione più semplice, facile da usare ed a più basso costo per la ricerca dei malfunzionamenti nei circuiti digitali. Generalmente si possono distinguere nelle seguenti categorie:

comparatori logici; utilizzano tecniche di comparazione per identificare nodi difettosi in un circuito digitale. Essi realizzano il test dinamico di un circuito integrato comparando le sue uscite con quelle di un identico integrato, di sicuro funzionamento, inserito di volta in volta sul comparatore stesso.

sonde logiche: rivelano i livelli logici in qualsiasi punto di un circuito attraverso l'accensione di uno o più diodi led.

clips logiche: sono indicatori dei livelli logici di più pin contemporaneamente mediante la loro inserzione diretta sul circuito integrato. Lo stato logico di ciascun piedino è mostrato da un led così da poter individuare la tabella della verità del particolare dispositivo sotto test.

sonde impulsive; questi tipi di sonde iniettano impulsi digitali tra le porte logiche, senza dover preventivamente procedere alla dissaldatura del componente. Esse automaticamente pilotano al livello logico basso i nodi che si trovano alti oppure pilotano alti i nodi che si trovavano bassi, senza introdurre sostanziali aumenti di corrente. Usato insieme ad un rilevatore

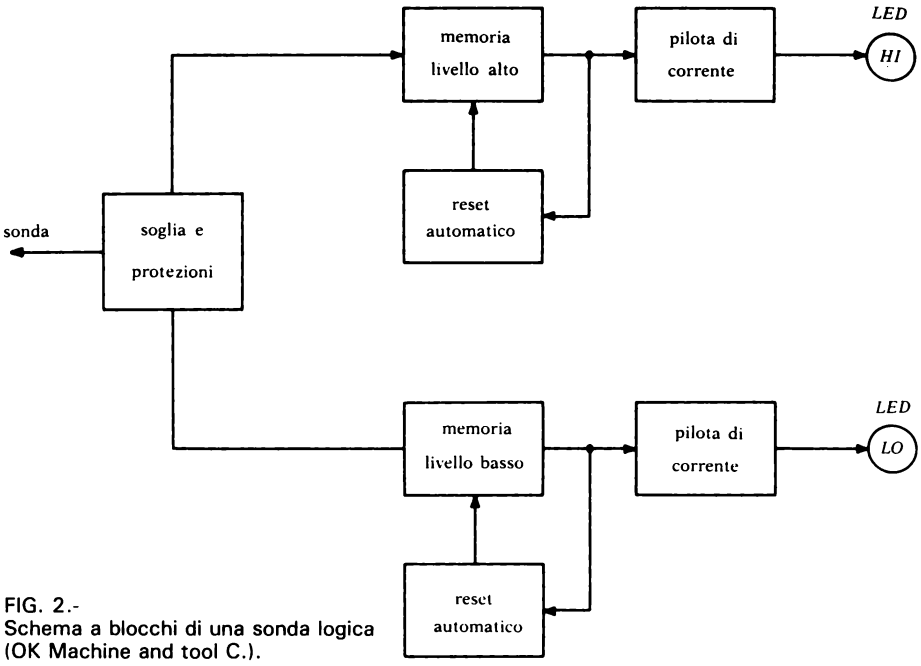


FIG. 2.-
Schema a blocchi di una sonda logica (OK Machine and tool C.).

digitale di corrente, può mostrare l'effettivo percorso della corrente in un complesso circuito.

Nella figura 2 è mostrato lo schema a blocchi di una sonda logica.

Per comodità, le soglie relative alle diverse famiglie (DTL, TTL, MOS, CMOS e microprocessori) sono programmate automaticamente allorché l'alimentazione della sonda logica è connessa a quella del circuito sotto test. Gli ingressi sconnessi di porte del tipo precedente, o il funzionamento in alta impedenza di porte tri-state, causano generalmente lo spegnimento dei led indicatori. La presenza delle memorie per i livelli alto e basso, permette di osservare impulsi dei due tipi anche se la loro frequenza è molto elevata poiché il reset automatico viene effettuato con un periodo molto più lungo (50 ÷ 100 ms).

Come esempio vengono mostrate alcune applicazioni di una sonda logica.

Isolamento di un difetto in una zona

In figura 3 è mostrato il metodo dello spostamento a metà che consiste nel seguente procedimento:

1) controllare la presenza del segnale in ingresso;
2) controllare la presenza del segnale in uscita. Se non è presente, scegliere un punto approssimativamente a metà;
3) se il segnale è presente nel punto 3, il difetto deve risiedere tra i punti 3 e 2.
4) controllare il punto 4 posto approssimativamente a metà tra i punti 3 e 2. Se il segnale non è presente, il malfunzionamento è tra i punti 3 e 4. Continuando in questo modo si riesce a localizzare il circuito integrato difettoso.

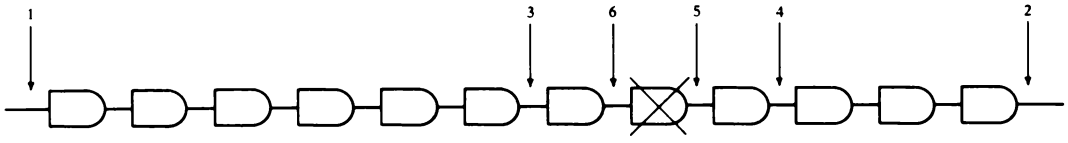


FIG. 3. - Metodo dello spostamento a metà per la localizzazione di un difetto di funzionamento.

Isolamento di una porta logica difettosa

Si supponga che nel circuito di figura 4 i segnali A e C siano giusti, mentre il segnale B non risponda appropriatamente.

La sonda logica potrebbe dare una delle sei seguenti indicazioni per il punto B:

1) il led HI rimane acceso
2) il led LO rimane acceso
3) il led HI non si accende mai ma si accende il led LO

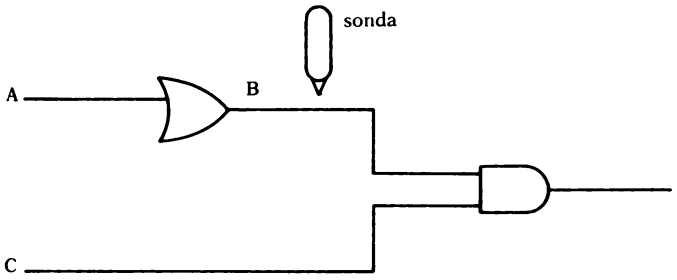


FIG. 4. - Individuazione di una porta logica difettosa.

4) il led LO non si accende mai ma si illumina il led HI.
5) nessuno dei due led risponde
6) il segnale nel punto B segue il segnale C.

In ciascuno dei casi precedenti, la soluzione consiste nella sostituzione dell'integrato, ma sarebbe possibile anche individuare la causa del difetto all'interno dell'integrato stesso.

3. Analizzatori logici

I malfunzionamenti che si incontrano nel dominio dei dati digitali si manifestano sotto forma di sequenze improprie di dati.

Indipendentemente dalla causa del problema, l'effetto è sempre funzionale, vale a dire che i dati non corretti vengono trasmessi anche quando la causa è dovuta al rumore o a picchi di tensione. Di conseguenza il primo passo nell'analisi consiste nel localizzare il guasto nella sequenza del flusso dei dati. La localizzazione di un malfunzionamento per mezzo di uno strumento esterno, richiede la registrazione dei dati o la sincronizzazione tra i due sistemi, seguita dalla cattura dei dati, dalla loro possibile manipolazione ed infine dalla visualizzazione su display.

Gli analizzatori logici si sono rivelati di grande utilità proprio nel difficile compito di accertare, in base a misure effettuate sui bus paralleli, quale parte del sistema non funziona correttamente. Da questo punto di vista le principali zone del sistema possono essere definite come: unità centrale, unità di controllo ed I/O o periferiche.

Il contenuto del bus della CPU può essere costituito da indirizzi, dati o istruzioni. Quello dell'I/0 invece, solitamente contiene dati asincroni in multiplex. Inoltre, in alcuni casi, la porta I/O può essere collegata ad una periferica fisicamente molto lontana e quindi ci si deve aspettare che i picchi di rumore ed i disturbi transitori, rappresentino un problema molto più serio su un bus di I/O che non su quello della CPU. Infine, i bus di

controllo sono simili alle linee di handshake, normalmente asincrone e sensibili ai disturbi transitori.

Gli analizzatori logici che si usano per tali scopi si dividono in tre categorie:
- analizzatori nel dominio dei dati;
- analizzatori nel dominio del tempo;
- generatori di trigger.

Per manipolare i dati del sistema sotto test, devono possedere i seguenti requisiti:

1) i dati devono essere letti in forma binaria. I livelli di soglia debbono essere i più vicini possibile a quelli della famiglia sotto test.

2) devono avere un numero sufficiente di ingressi in modo da monitorare l'intera lunghezza della parola in parallelo.

3) i dati che fluiscono sulle linee, devono essere letti dall'analizzatore nello stesso modo in cui sono letti dal sistema sotto test. In molte applicazioni i dati iniziano a cambiare subito dopo la transizione del clock e diventano stabili per breve tempo prima dell'arrivo del successivo fronte di clock. Ciò richiede che l'analizzatore abbia un tempo th (hold-time) e di assestamento (set-up) i più piccoli possibile.

4) non deve influenzare il sistema sotto test.

In figura 5 è mostrato lo schema a blocchi di un analizzatore logico.

Nel blocco di controllo vengono impostate dall'operatore le condizioni che specificano il test, come ad esempio:

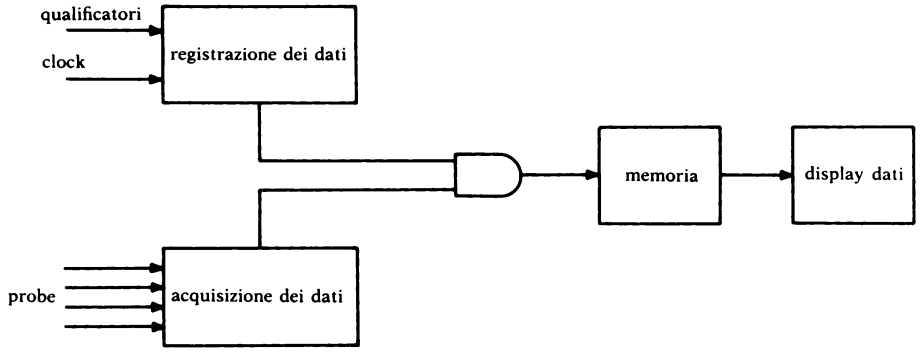


FIG. 5. - Percorso dei dati e dei controlli in un analizzatore logico.

a) la parola di trigger che selezioni una particolare parola, in modo tale che quando quest'ultima coincide con quella impostata, abbia inizio o termine la memorizzazione dei dati;

b) la velocità del clock. Essa determina la massima velocità dello strumento sotto test, che può essere manipolata dall'analizzatore.

Nell'analisi di stato, la velocità di clock è la massima permissibile velocità di ripetizione di impulsi dell'apparecchio sotto test. Nell'analisi nel tempo, è la velocità alla quale vengono presi campioni dell'onda in ingresso.
c) informazioni circa i fronti d'onda, la logica, l'eventuale cattura di disturbi (glitch), ecc.

Il blocco di acquisizione risolve problemi di tipo meccanico ed elettrico circa il collegamento fisico tra sonda e linea da esaminare.

Una volta avvenuta la coincidenza tra i dati impostati dall'operatore e quelli fluenti sulla linea, ha inizio la loro memorizzazione e visualizzazione su display.

Analizzatore di temporizzazioni

Una delle funzioni principali di questi analizzatori è di mostrare le relazioni temporali tra i segnali di un circuito logico.

Essi non sostituiscono i voltmetri digitali o gli oscilloscopi, ma fornisco, invece, differenti tipi di misura:
- mostrano l'attività di più canali simultaneamente (da otto a ventiquattro e più canali);
- prelevano informazioni in modo asincrono o sincrono, un filtro temporale distingue i dati validi da un'attività casuale;
- possono memorizzare dati che accadono prima della parola di trigger selezionata;
- possono ritardare l'inizio della memorizzazione di un tempo variabile;
- possono catturare impulsi spuri (glitches).

L'esibizione di informazioni temporali avviene nella forma idealizzata mostrata in figura 6.

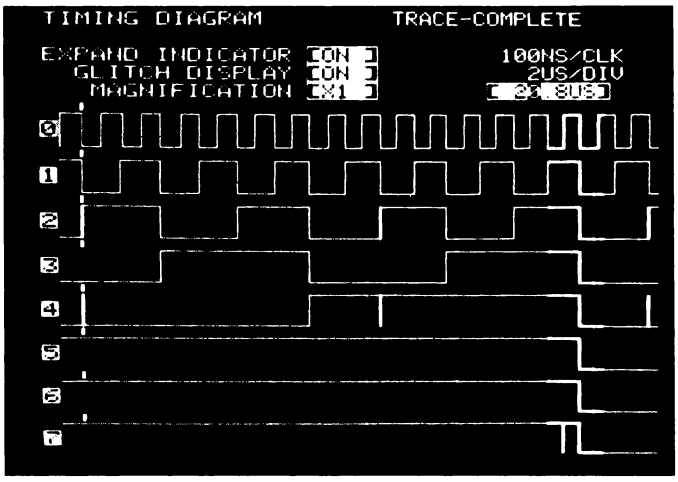


FIG. 6. - Esibizione degli stati (alti o bassi) su ciascuna delle otto linee testate in concomitanza di un clock interno (Hewlett-Packard).

La cattura dei glitches, il rilevamento di temporizzazioni non appropriate o le sequenze errate di parole sono il principale campo di impiego di tali analizzatori.

Nella figura 7 sono mostrate due diverse forme di rivelazione dei glitches.

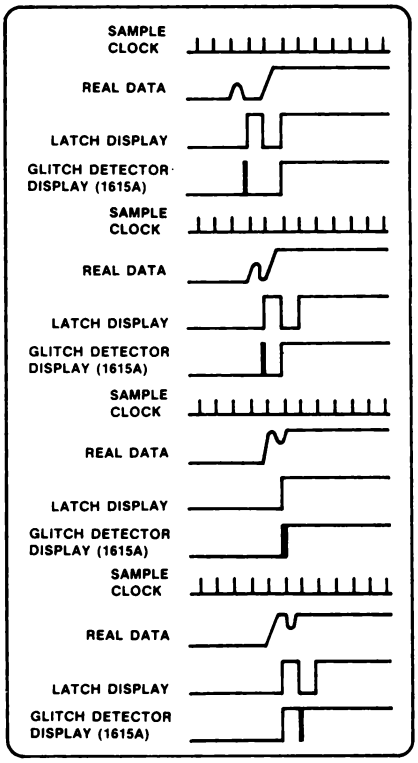


FIG. 7. - Cattura dei glitches mediante latch o mediante memoria separata (Hewlett-Packard).

Come si vede la cattura mediante latch mostra la stessa forma d'onda se il glitch avviene due periodi prima, un periodo prima o un periodo di clock dopo la transizione del dato, senza possibilità di distinzione.

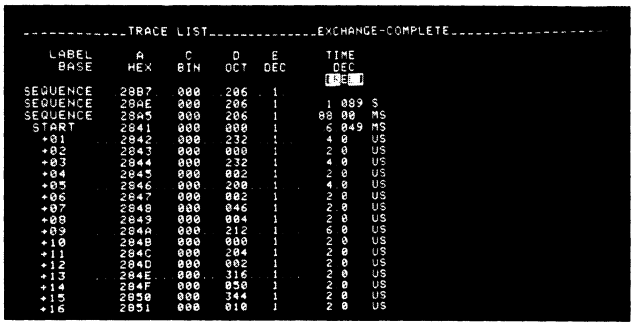
Analizzatore di stati logici

Gli analizzatori di stati logici catturano e mostrano il flusso di eventi che avvengono in modo sincrono, in un sistema logico.

La visione dell'attività del sistema avviene mediante la lista degli stati sullo schermo dell'analizzatore.

Gli analizzatori più sofisticati offrono la possibilità di controllare 24 o 36 linee in modo parallelo. Anche i formati tramite i quali vengono mo-

strati i dati può cambiare grandemente potendo andare dalla semplice lista di 0 e di 1 al riassemblamento del particolare linguaggio mnemonico usato da un particolare microprocessore. Nella figura 8 è mostrata la lista di una sequenza di eventi che avvengono in sincronismo con il segnale di clock del sistema in esame.



TRACE LIST EXCHANGE-COMPLETE

LABEL BASE	A HEX	C BIN	D OCT	E DEC	TIME DEC
SEQUENCE	28B7	000	206	1	
SEQUENCE	28AE	000	206	1	1 009 S
SEQUENCE	28A5	000	206	1	88 00 MS
START	2841	000	000	1	6 049 MS
+01	2842	000	232	1	4 0 US
+02	2843	000	000	1	2 0 US
+03	2844	000	232	1	4 0 US
+04	2845	000	002	1	2 0 US
+05	2846	000	200	1	4 0 US
+06	2847	000	002	1	2 0 US
+07	2848	000	046	1	2 0 US
+08	2849	000	004	1	2 0 US
+09	284A	000	212	1	6 0 US
+10	284B	000	000	1	2 0 US
+11	284C	000	204	1	2 0 US
+12	284D	000	002	1	2 0 US
+13	284E	000	316	1	2 0 US
+14	284F	000	050	1	2 0 US
+15	2850	000	344	1	2 0 US
+16	2851	000	010	1	2 0 US

FIG. 8. - Flusso dei dati in un analizzatore di stati logici (Hewlett-Packard).

ESEMPIO

Nel seguente esempio viene descritto lo schema a blocchi ed il funzionamento di un semplice analizzatore di stati logici che può essere realizzato in un laboratorio. (fig. 9).

Il funzionamento è il seguente:

I dati, prelevati dal bus che si vuole analizzare, vengono controllati tramite otto buffer (blocco A) che sono abilitati tramite un pulsante di start; ciò per non permettere l'afflusso continuo al blocco comparatore (blocco B) di configurazioni di bit non desiderate. Quando i buffer vengono abilitati, i dati fluiscono nel comparatore (blocco B), dove è già stata impostata manualmente la parola di trigger tramite interruttori posizionati al livello logico zero o uno.

Man mano che le configurazioni di dati sul bus cambiano, all'interno del comparatore viene effettuata l'operazione per verificare quando avviene l'uguaglianza tra la parola di trigger ed una configurazione dei bit in ingresso. Nel caso che nessuna configurazione coincida con quella impostata, il sistema rimane inattivo, mentre ad uguaglianza verificata, viene emesso un segnale T che, inviato ad un flip-flop facente parte della logica di controllo (blocco C), memorizza lo stato del segnale T e provvede ad abilitare l'inizio del conteggio dei contatori (abilitazione tramite $\overline{EN}$ del blocco D).

La funzione dei contatori è quella di fornire gli indirizzi alla memoria che deve immagazzinare tutte le configurazioni di bit del bus che si presentano dalla parola di trigger in poi. Naturalmente occorre scegliere il nume-

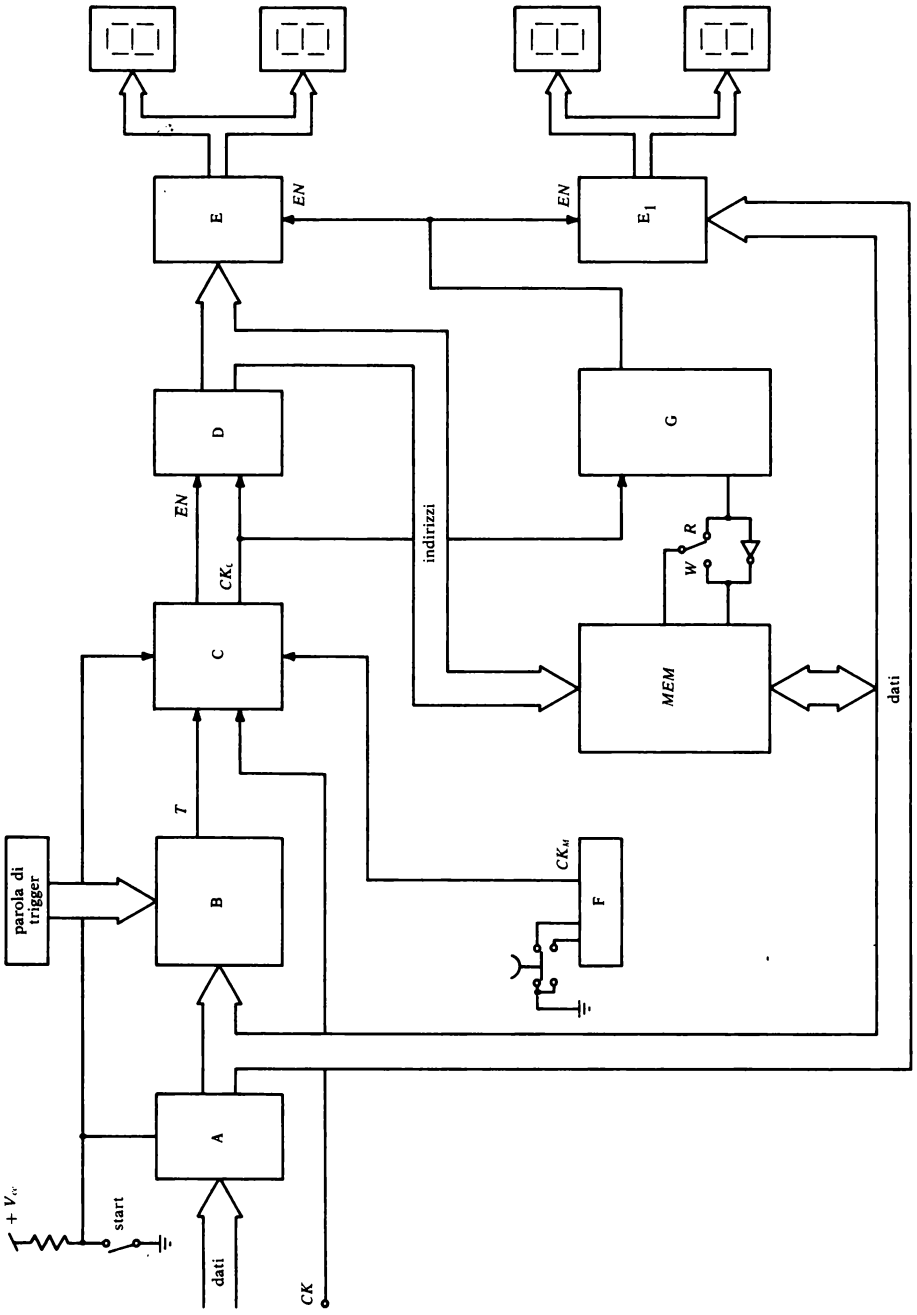


FIG. 9. - Schema a blocchi di un analizzatore di stati logici A) buffer di ingresso; B) comparatore; C) logica di controllo; D) contatori; E) latch e decodiche; F) clock manuale per la lettura; G) temporizzazione per la memoria; CKc) clock emesso dalla logica di controllo; T) segnale di uguaglianza verificata.

ro massimo di parole che si vuole memorizzare, per dimensionare i contatori e la memoria stessa che deve avere un parallelismo pari a quello del bus sotto test.

Gli indirizzi vengono inoltre inviati ai blocchi di decodifica E che permettono, mediante display a sette segmenti, la loro visualizzazione.

Al termine del conteggio, cioè quando si è raggiunta la massima capacità della memoria, il contatore emette un segnale che viene usato per disabilitarsi ed azzerarsi.

Il blocco G contiene porte logiche la cui unica funzione è di ripristinare le corrette temporizzazioni tra dati, indirizzi e controlli in arrivo alla memoria per realizzare una corretta scrittura.

La lettura dei dati immagazzinati viene effettuata disabilitando i buffer di ingresso ed il clock del sistema, mediante il tasto start, in modo da impedire un'ulteriore afflusso di dati e successivamente introducendo un clock manuale (blocco F). In questo modo si possono incrementare i contatori di un passo alla volta per permettere la lettura delle configurazioni di bit residenti nella memoria. La loro visualizzazione avviene tramite il blocco E_1 di decodifiche ed i display a sette segmenti.

4. Analizzatori di firma

Un analizzatore di firma (signature analyzer) cattura e mostra l'unica firma associata ad un determinato nodo in un circuito sotto test. Comparando la firma rilevata, con quella corretta, si può risalire al nodo causa di errori.

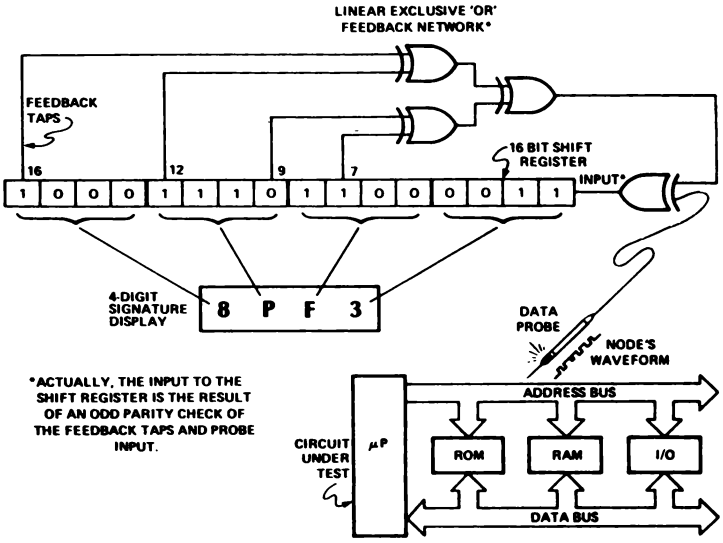


FIG. 10. - Principio di funzionamento di un analizzatore di firma (Hewlett-Packard).

Le principali caratteristiche che li distinguono uno dall'altro sono:
1) il tipo di reazione interna sui registri;
2) i caratteri mostrati sul display;
3) le caratteristiche di selezione dei segnali;
4) le soglie logiche di ingresso.

La figura 10 mostra il modello per i 16 registri interni della firma comprendente le quattro sorgenti di reazione, adottato dalla H.P. In essa si vede come l'analizzatore comprima le forme d'onda rilevate in un nodo durante un ciclo di misura.

Ciascun raggruppamento di 4 bit concorre a formare la firma secondo una ben determinata tabella di corrispondenza che, molte volte, non corrisponde alla numerazione standard esadecimale. Ad esempio quella adottata dalla H.P. è la seguente:

bit	display
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	C
1100	F
1101	H
1110	P
1111	U

Il nuovo set di caratteri elimina le possibili confusioni tra la lettera minuscola b ed il numero 6, o la lettera maiuscola B ed il numero 8.

Il ciclo di misura di un analizzatore di firma è controllato per mezzo di una porta. Quando essa è aperta, l'analizzatore preleva una nuova firma tramite la sonda dei dati, che viene poi inviata ai display non appena la porta viene chiusa.

Il funzionamento della porta è controllato in vari modi. Nella figura 11 è mostrato quello utilizzato dalla H.P. che utilizza i tre segnali: START, STOP e CLOCK.

Il segnale di clock è utilizzato per sincronizzare l'analizzatore al congegno sotto test. I livelli logici della forma d'onda misurata dalla sonda sono campionati sui fronti di clock ed ignorati in tutti gli altri istanti. Tali livelli

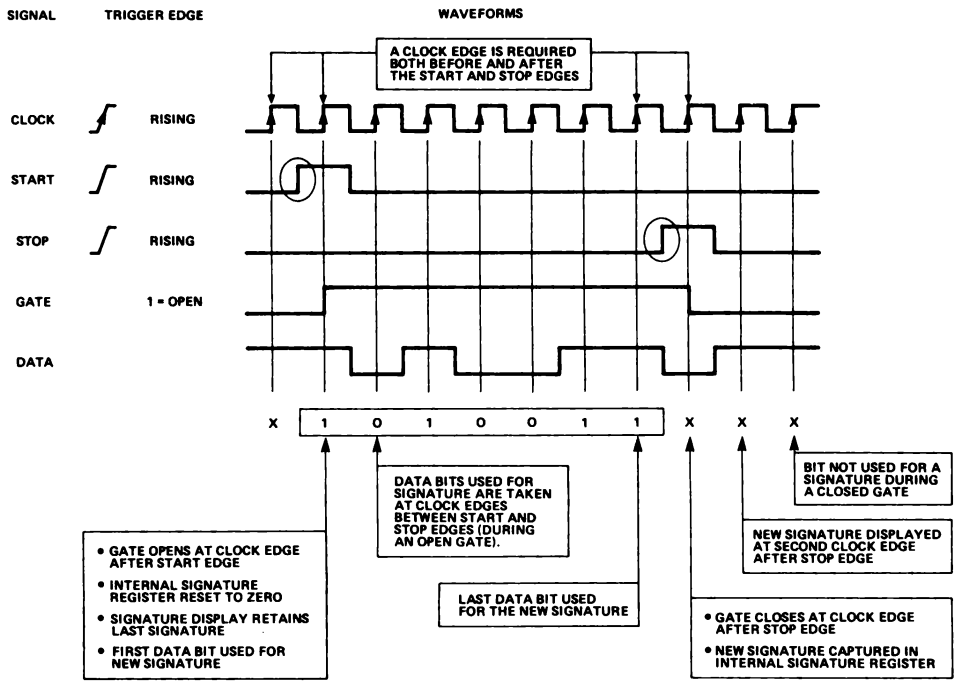


FIG. 11. - Controllo e funzionamento della porta (Hewlett-Packard).

logici sono usati come bit di dati da comprimere per la formazione della firma. Gli ingressi di START e di STOP sono anch'essi rilevati in modo sincrono con il clock: la porta viene aperta sul fronte di clock immediatamente seguente il fronte del segnale di START ed analogamente, la porta viene chiusa in corrispondenza al fronte di clock seguente il segnale di STOP.

Ci sono molti modi differenti di controllare il funzionamento della porta di un analizzatore di firma. Per esempio, START e STOP possono essere connessi allo stesso o a differenti segnali, così come possono essere triggerati su fronti differenti.

Usualmente lo START apre la porta, mentre lo STOP la chiude, ma uno solo di essi può commutarla in modo alternativo. La sua apertura può essere più o meno lunga ed il primo e l'ultimo bit possono essere usati in modi differenti.

Nella figura 12 è mostrato come possono essere generati i tre segnali di controllo in una situazione reale.

Lo START e lo STOP sono connessi ad un bit prelevato da un latch indirizzato come una porta di I/0. Il primo è triggerato sul fronte in salita, mentre il secondo su quello in discesa.

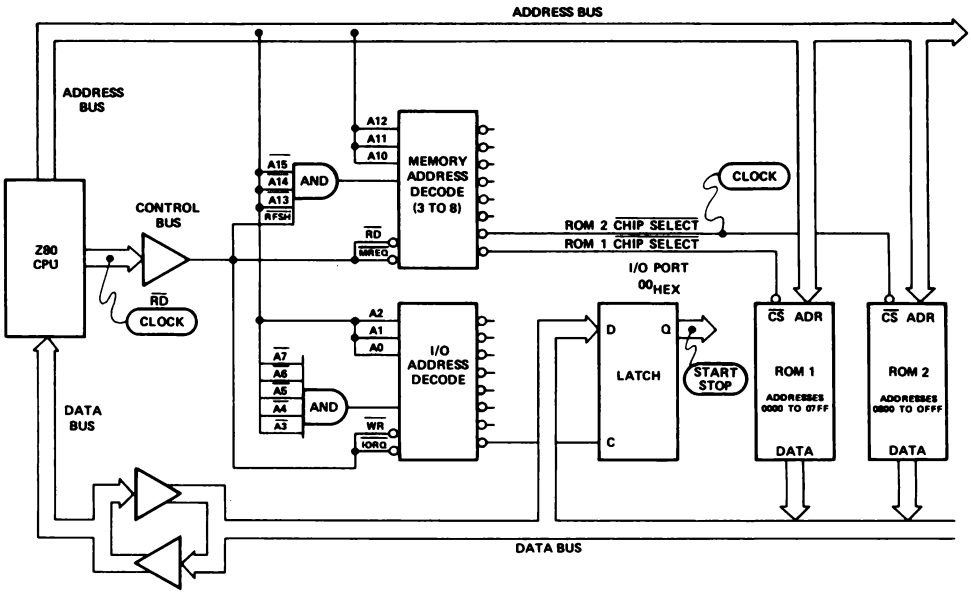


FIG. 12. - Generazione dei segnali di controllo della porta in un sistema a μP (Hewlett-Packard).

Tali fronti sono generati tramite la combinazione di questo hardware con l'esecuzione di un opportuno programma di stimolo immagazzinato nella ROM 1. Ciò permette all'analizzatore di firma di controllare la correttezza della ROM 2 e dei circuiti ad essa associati.

Il programma consiste semplicemente nella lettura di tutte le locazioni di memoria della ROM 2. Prima della lettura della prima locazione di memoria, è necessario che il programma setti un bit nel latch per aprire la porta dell'analizzatore e successivamente tutti i dati letti sul data bus sono usati per comporre la firma. Dopo la lettura, di tutte le locazioni della ROM 2, il programma resetta il bit per chiudere la porta e ricomincia per l'esecuzione di un nuovo ciclo.

Se come clock per l'analizzatore, viene usato il segnale $\overline{RD}$ dello Z 80, i fronti di START e di STOP vengono rilevati correttamente facendo aprire e chiudere la porta normalmente.

La durata dell'apertura della porta deve essere tale da garantire il prelevamento di tutti gli stati funzionali di quel nodo in modo che l'analizzatore, comprimendo tutti i dati che lo attraversano, mostri una firma che dia indicazioni circa il corretto funzionamento del dispositivo che lo pilota.

È necessario che il periodo del clock di campionamento degli stati funzionali del nodo, sia tale da campionarli tutti, altrimenti può accadere che non venga rilevato un errore di un singolo bit, come è mostrato nella figura 13.

Errori di più bit corrispondono alla variazione della firma con una probabilità di almeno il 99,99% se i bit sono stati correttamente campionati e se il congegno è stato eccitato in tutte le sue funzioni.

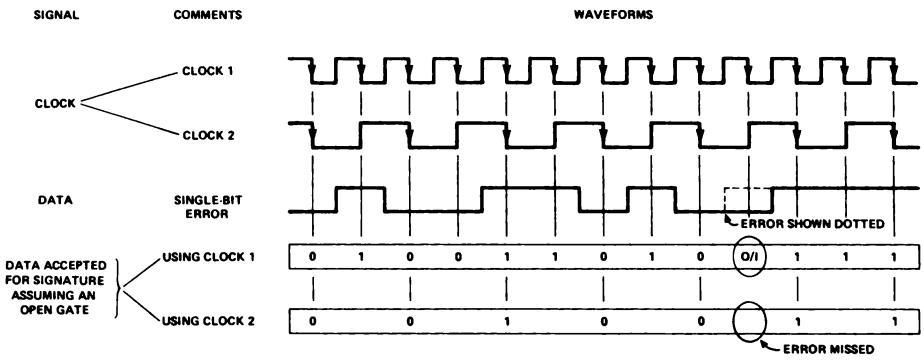


FIG. 13. - La scelta di un clock di frequenza non adatta, dà origine ad errori di lettura.

Firme corrette od errate

Sono definite corrette quelle firme che corrispondono a quelle documentate nelle procedure di controllo di un prodotto e che sono fornite dalle case costruttrici. Il rilevamento di una firma errata indica una forma d'onda errata per quel nodo e permette di risalire indietro nel circuito fino al ritrovamento del particolare congegno che genera una firma sbagliata (e quindi stati di uscita errati) pur ricevendo in ingresso firme corrette.

Naturalmente è necessario che i creatori degli stimoli e delle firme, garantiscano la ripetibilità della stessa firma per quello stesso nodo se:
1) sono presi nodi identici nella stessa apparrecchiatura;
2) sono presi nodi corrispondenti in prodotti uguali;
3) sono effettuate identiche misure di firma;
4) sono usati programmi di stimolo identici;
5) sono usati analizzatori con caratteristiche compatibili.

5. Sistemi di sviluppo

Nel progetto di un prodotto basato sui microprocessori, la procedura ideale per la realizzazione e lo sviluppo del prototipo richiede l'integrazione continua fra software ed hardware in modo tale che il secondo possa essere eccitato e provato dal primo e viceversa, man mano che si procede. In caso contrario, il non corretto funzionamento dovuto alla imperfetta integrazione dei due componenti, può portare alla completa ridefinizione

del progetto con notevoli perdite di tempo e di risorse.

Una tale integrazione è praticamente impossibile da ottenere fin dallo stato iniziale se non attraverso l'utilizzazione di un sistema di sviluppo che è in grado di sostituirsi alla parte mancante di un tipo o dell'altro.

Il suo impiego consiste generalmente nella esecuzione di una delle seguenti attività:

1) compilazione o assemblaggio di un programma;
2) edizione di un programma;
3) stimolazione delle memorie;
4) debugging sia software che hardware;
5) emulazione.

Ovviamente nulla esclude che i sistemi di sviluppo possano differire uno dall'altro per molteplici altre particolarità che ciascuna casa costruttrice ha inserito.

Prima di procedere ad una pur sommaria descrizione del loro principio di funzionamento, demandando ai manuali il compito di una maggiore precisione, è necessario definire correttamente il significato di alcuni termini di maggiore ricorrenza.

Programma sorgente

Viene chiamato in questo modo qualsiasi programma mnemonico, scritto dal programmatore in un linguaggio più o meno evoluto.

Programma oggetto

È quello che viene ottenuto traducendo il programma sorgente in un linguaggio direttamente comprensibile dall'elaboratore (successione di parole formate da 0 ed 1).

Programma assemblatore

Mediante l'uso del programma assemblatore, un computer traduce un linguaggio a livello Assembly (programma sorgente), in un insieme di 0 e di 1 (programma oggetto) direttamente comprensibile della macchina. È anche in grado di rivelare eventuali errori di sintassi nel programma sorgente. Nei primi elaboratori, il programmatore era costretto ad introdurre il programma fornendo manualmente i bit uno alla volta.

Programma compilatore

I compilatori sono programmi che hanno scopi simili agli assemblatori ma sono più orientati verso il flow-chart e l'uomo, vale a dire possono tradurre in 0 ed 1 i linguaggi ad alto livello ed anche produrre programmi in linguaggio Assembly. Sul mercato hanno avuto un discreto successo i compilatori PL/M che producono codici per i più diffusi microprocessori.

In definitiva un compilatore, rispetto ad un assemblatore, ha il vantag-

gio di non dover programmare ripetutamente anche i più piccoli passaggi, ma in tal modo si perde quasi totalmente il controllo necessario se si vuol sfruttare al massimo le risorse del sistema.

Programma interprete

È un programma che permette di far eseguire dall'elaboratore le istruzioni date in un linguaggio ad alto livello, senza dover passare attraverso l'assemblaggio o la compilazione.

Tali operazioni vengono eseguite internamente come passaggio intermedio senza l'intervento dell'operatore. Diminuendo il numero dei passaggi rende il processo maggiormente interattivo.

Programma monitor

Questo programma, chiamato anche sistema operativo, sovrintende al funzionamento di tutto il sistema. Negli elaboratori di piccole dimensioni entra automaticamente in funzione all'atto dell'accensione, mentre nei più grandi deve essere caricato da una memoria di massa esterna (solitamente dischi). La sua entrata in funzione rende possibile l'accettazione di comandi esterni che causano la esecuzione dei diversi programmi e, non appena questi ultimi sono terminati, il controllo torna al sistema operativo.

Programma editor

Il programma editor rende possibile cambiare, cancellare, inserire caratteri o righe nuove nella stesura, tramite tastiera, di un programma sorgente.

Emulazione

L'emulazione consiste nell'imitare un sistema con un altro tale che il sistema imitante accetti gli stessi dati, esegua gli stessi programmi e raggiunga gli stessi risultati del sistema imitato.

Programma linker

Permette di ricollegare, in modo omogeneo, parti di programma realizzate in modo frammentario, ottenendo un unico programma oggetto.

Programma di debugger

Il debugger è un programma utilizzato per verificare la corretta esecuzione del programma oggetto, da parte del sistema in esame. Nelle operazioni di debugging si può far iniziare e terminare l'esecuzione del programma, in funzione di determinate parole o eventi e stampare il contenuto di particolari locazioni di memoria per facilitare la correzione.

Nella tabella 1 è mostrata in forma compatta una semplice suddivisione funzionale dei linguaggi e dei programmi.

livello del linguaggio	tradotto mediante	immagazzinato in
microistruzioni	logica cablata	porte e registri
linguaggi macchina	microcodici	memorie veloci
linguaggi assembly	assembler	memoria principale
linguaggi evoluti	compilatori, interpreti	memoria principale

Tabella 1. - Linguaggi di programmazione.

Funzionamento di un sistema di sviluppo

All'atto dell'accensione il primo passo effettuato dal sistema di sviluppo, è l'esecuzione del particolare programma chiamato *monitor* o *sistema operativo* (secondo il costruttore).

Il compito del monitor è quello di dialogare con l'operatore, tramite una tastiera o un terminale video, per rispondere ai comandi da esso impartiti.

Normalmente si inizia richiamando dalla memoria di massa del sistema di sviluppo (dischi), il programma *editor* che permette all'operatore la creazione di un nuovo programma sorgente. Tramite il programma di editor, è possibile la scrittura, la correzione, l'impaginazione del nuovo programma ed il suo successivo caricamento in memoria. Nella tabella 2 è mostrata ad esempio la lista dei comandi dell'editor dell'8550 Tektronix.

Successivamente viene richiamato il programma *assemblatore* per tradurre il programma sorgente in oggetto. L'assemblatore opera sia sui simboli che rappresentano i codici operativi che su quelli che rappresentano indirizzi simbolici. I primi sono fissi e stabiliti dal particolare set di istruzioni, mentre i secondi sono nomi arbitrari che indicano posizioni di memoria contenenti dati. Alcuni codici operativi, anzichè costituire istruzioni nel programma oggetto da generare, sono degli ordini dati all'assemblatore stesso ed in tal caso vengono chiamati pseudo-istruzioni. Sono esempi di pseudo-istruzioni: ORG, DATA, ecc.

Un'altra importante possibilità è quella fornita da un macroassemblatore che permette di dare un nome ad una sequenza di più istruzioni in modo da richiamarle tutte insieme con la sola scrittura del nome. A differenza delle chiamate di subroutine, le macro vengono più spesso usate nel controllo delle sequenze di istruzioni per l'I/0 e le amplia se necessario.

Poiché l'uso delle macro fa occupare un numero molto maggiore di byte di memoria, nei microcalcolatori vengono utilizzate con limitazione.

Command Function	Command Function
A — Advance lines	**Q** — Query (verify command execution)
C — Change characters	**(pd)** — Page down
(cd) — Cursor down	**(pu)** — Page up
(ch) — Cursor home	**R** — Replace characters
(cl) — Cursor left	**(rm)** — Revise mode
(cr) — Cursor right	**S** — Stop editor, don't save results
(cu) — Cursor up	**(sd)** — Scroll down
(dc) — Delete characters	**(sl)** — Scroll left
(dl) — Delete lines	**(sr)** — Scroll right
EC — Echo commands during command file execution	**(su)** — Scroll up
EP — Editor profile	**T** — Tab stop definition
ER — Edit read	**U** — Undelete characters
EW — Edit write	**V** — View lines (display non-printing characters)
EX — Exit editor, save results	**WE** — Define ESC wildcard character
F — Find character string	**WG** — Define general wildcard character
H — Display command syntax	**WI** — Define case-ignore wildcard
(ic) — Insert characters	**XL** — Exchange with lower case
(il) — Insert lines	**XU** — Exchange with upper case
J — Jump characters	
MD — Macro definition	
ML — Macro list	
MX — Macro execute	
O — Open edit read/write file	

Note: Commands enclosed in ( ) symbols are user configurable and may be assigned to a specific key.

Tabella 2. - Lista dei comandi dell'editor dell'8550 (Tektronix).

Un comando tipico per richiedere l'assemblaggio è ad esempio ASMB seguito dal nome del programma sorgente e da quello che si vuole dare al programma oggetto. A causa del continuo scambio di informazioni con il disco, nei programmi molto lunghi sono necessarie anche varie decine di minuti per eseguire l'operazione. Se vengono rilevati degli errori, è necessario correggerli richiamando il programma editor e riassemblando successivamente fino ad ottenere un perfetto assembly.

Per risparmiare tempo in queste operazioni, se l'assembler è in grado di produrre codici oggetto rilocabili, il programma può essere suddiviso in vari moduli di più facile e veloce manipolazione. Spetta poi ad un programma *linker* riunire i moduli separati fornendone uno unico allocato in posizioni di memoria contigue.

Una volta terminata la fase di assemblaggio, si può passare al *debug-*

ging, vale a dire all'esecuzione passo-passo del programma per la ricerca delle imperfezioni (letteralmente: bugs = pulci) e della loro eliminazione. Il programma che sovrintende al debug permette l'interazione della tastiera e del display, direttamente con il contenuto dei registri e delle locazioni di memoria in modo da poter osservare il risultato delle loro variazioni. Per fare ciò esso traduce automaticamente il linguaggio macchina in assembly e viceversa senza dover riassemblare il programma ogni volta che vi sono variazioni.

Un'ulteriore opzione che serve a risparmiare tempo durante il debugging, è la possibilità di sezionare l'esecuzione del programma mediante l'inserzione di *breakpoint.* In questi casi l'esecuzione viene bloccata all'indirizzo in cui è inserito il breakpoint, in modo da controllare solo la parte che interessa.

Quando il software è stato messo a punto, si può procedere alla verifica della perfetta interazione con l'hardware mediante una fra le più potenti opzioni del sistema di sviluppo: l'emulazione *in-circuit* oppure la simulazione dell'hardware.

L'emulazione in tempo reale usa un processore di emulazione funzionalmente identico a quello impiegato nel prototipo completo. Durante lo sviluppo software, il programma è eseguito direttamente sull'emulatore, con le funzioni di I/0 del prototipo simulate tramite uno speciale software. Durante l'integrazione, una sonda multipla connette il processore di emulazione al posto di quello, lasciato libero, sul prototipo. L'emulatore, sotto il controllo del debug, può allora usare l'effettivo hardware sotto test e seguirne l'interazione con il software.

Sul mercato esistono sistemi di sviluppo dedicati ai particolari microprocessori e quelli di uso universale. In questi ultimi sono forniti compilatori ed emulatori per i microprocessori più diffusi. I moduli di emulazione in tempo reale, sono generalmente costituiti da una scheda di controllo e da una sonda specializzata per il collegamento con il processore in emulazione.

La scelta fra un sistema di sviluppo dedicato od uno universale è legata ovviamente a criteri di prezzo ed alle particolari esigenze dell'utilizzatore.

CAPITOLO DECIMO

APPLICAZIONI DEL SISTEMA MINIMO Z80

Nel paragrafo 12 del quarto capitolo è stato esaminato un dispositivo hardware-software in grado di collaudare circuiti integrati (C.I.) mediante il metodo *tabulare*. Tale metodo, nel caso che gli integrati sotto esame siano del tipo MSI, molte volte non è applicabile in quanto il software deve essere scritto di volta in volta riferendosi alla particolare funzione svolta dall'integrato sotto test.

Nelle pagine seguenti saranno esaminati l'hardware ed il software necessari per il collaudo di due decodificatori della serie 74 (Texas) e cioè l'SN 7446 e l'SN 7442. Sarà inoltre affrontato il progetto e la realizzazione di un *sistema minimo* in grado di eseguire una assegnata funzione utilizzando sia porti di I/0 tradizionali sia il PIO Z80, componente quest'ultimo ampiamente trattato nel quarto capitolo.

1. Test di collaudo del circuito integrato SN 7446

L'integrato SN 7446 è un decoder MSI open-collector BCD-sette segmenti la cui piedinatura e tabella funzionale sono riportate nelle fig. 1 e 2.

Dall'esame della tabella funzionale si può notare che il pin RBO dell'integrato può essere assunto sia come ingresso che come uscita, in questo esempio sarà considerato come uscita.

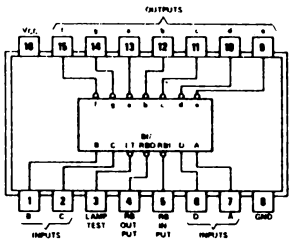
La tabella funzionale dell'integrato non ha una logica e di conseguenza è necessario inviare delle configurazioni d'ingresso sequenziali e in corrispondenza di queste leggere le esatte uscite memorizzate in una tabella e confrontarle poi con quelle effettive dell'integrato sotto test segnalando il risultato del confronto in un display.

Si inizia dalla configurazione 1, facente capo alla seconda riga della tabella, nella quale il valore di RBI risulta indifferente e si continua fino alla configurazione 15 per poi testare la configurazione 0 per la quale RBI è forzato al livello logico alto.

BCD-TO-SEVEN-SEGMENT DECODERS/DRIVERS

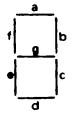
46 ACTIVE-LOW, OPEN-COLLECTOR, 30-V OUTPUTS

47 ACTIVE-LOW, OPEN-COLLECTOR, 15-V OUTPUTS



SN5446A (J, W) **SN7446A (J, N)**
SN54L46 (J) **SN74L46 (J, N)**
SN5447A (J, W) **SN7447A (J, N)**
SN54L47 (J) **SN74L47 (J, N)**
SN54LS47 (J, W) **SN74LS47 (J, N)**

See page 7-22



SEGMENT IDENTIFICATION

NUMERICAL DESIGNATIONS AND RESULTANT DISPLAYS

'46A, '47A, 'L46, 'L47, 'LS47 FUNCTION TABLE

DECIMAL OR FUNCTION	INPUTS						BI/RBO†	OUTPUTS							NOTE
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g	
0	H	H	L	L	L	L	H	ON	ON	ON	ON	ON	ON	OFF	1
1	H	X	L	L	L	H	H	OFF	ON	ON	OFF	OFF	OFF	OFF	
2	H	X	L	L	H	L	H	ON	ON	OFF	ON	ON	OFF	ON	
3	H	X	L	L	H	H	H	ON	ON	ON	ON	OFF	OFF	ON	
4	H	X	L	H	L	L	H	OFF	ON	ON	OFF	OFF	ON	ON	
5	H	X	L	H	L	H	H	ON	OFF	ON	ON	OFF	ON	ON	
6	H	X	L	H	H	L	H	OFF	OFF	ON	ON	ON	ON	ON	
7	H	X	L	H	H	H	H	ON	ON	ON	OFF	OFF	OFF	OFF	
8	H	X	H	L	L	L	H	ON	ON	ON	ON	ON	ON	ON	
9	H	X	H	L	L	H	H	ON	ON	ON	OFF	OFF	ON	ON	
10	H	X	H	L	H	L	H	OFF	OFF	OFF	ON	ON	OFF	ON	
11	H	X	H	L	H	H	H	OFF	OFF	ON	ON	OFF	OFF	ON	
12	H	X	H	H	L	L	H	OFF	ON	OFF	OFF	OFF	ON	ON	
13	H	X	H	H	L	H	H	ON	OFF	OFF	ON	OFF	ON	ON	
14	H	X	H	H	H	L	H	OFF	OFF	OFF	ON	ON	ON	ON	
15	H	X	H	H	H	H	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF	
BI	X	X	X	X	X	X	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	2
RBI	H	L	L	L	L	L	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	3
LT	L	X	X	X	X	X	H	ON	ON	ON	ON	ON	ON	ON	4

H = high level, L = low level, X = irrelevant

NOTES:
1. The blanking input (BI) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input (RBI) must be open or high if blanking of a decimal zero is not desired.
2. When a low logic level is applied directly to the blanking input (BI), all segment outputs are off regardless of the level of any other input.
3. When ripple-blanking input (RBI) and inputs A, B, C, and D are at a low level with the lamp test input high, all segment outputs go off and the ripple-blanking output (RBO) goes to a low level (response condition).
4. When the blanking input/ripple blanking output (BI/RBO) is open or held high and a low is applied to the lamp-test input, all segment outputs are on.

†BI/RBO is wire-AND logic serving as blanking input (BI) and/or ripple-blanking output (RBO).

FIG. 1 - Piedinatura e tabella funzionale dell'SN 7446 (Texas)

FUNCTION TABLE

INPUTS						OUTPUTS															
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = high level, L = low level, X = irrelevant

FIG. 2 - Tabella funzionale dell'SN 74154

Le configurazioni di uscita della tabella sette segmenti (a,b,c,d,e,f,g) più l'uscita RBO, con RBO variabile binaria a peso più alto, sono le seguenti:

RBO	g	f	e	d	c	b	a	
1	1	1	1	1	0	0	1	riga 2 (esadecimale F9)
1	0	1	0	0	1	0	0	riga 3 (esadecimale A4)
—	—	—	—	—	—	—	—	
1	1	1	1	1	1	1	1	riga 15 (esadecimale FF)

dove si è indicato con 1 lo stato di OFF e con 0 lo stato di ON.

Le quindici configurazioni di uscita ricavate dalla tabella sopra riportata valgono in esadecimale:

Tab: F9, A4, B0, 99, 92, 83, 78, 80, 98, A7, B3, 9D, 96, 87, FF.

Per queste configurazioni RBI è indifferente mentre per la prima configurazione (riga 0) RBI vale 1. Il *Lamp-Test* (LT) se portato basso deve accendere tutti i segmenti del display.

L'hardware del circuito, mostrato in fig. 3, utilizza due porti di uscita (SN 74374), un porto di ingresso (SN 74244), un SN 74244 utilizzato come

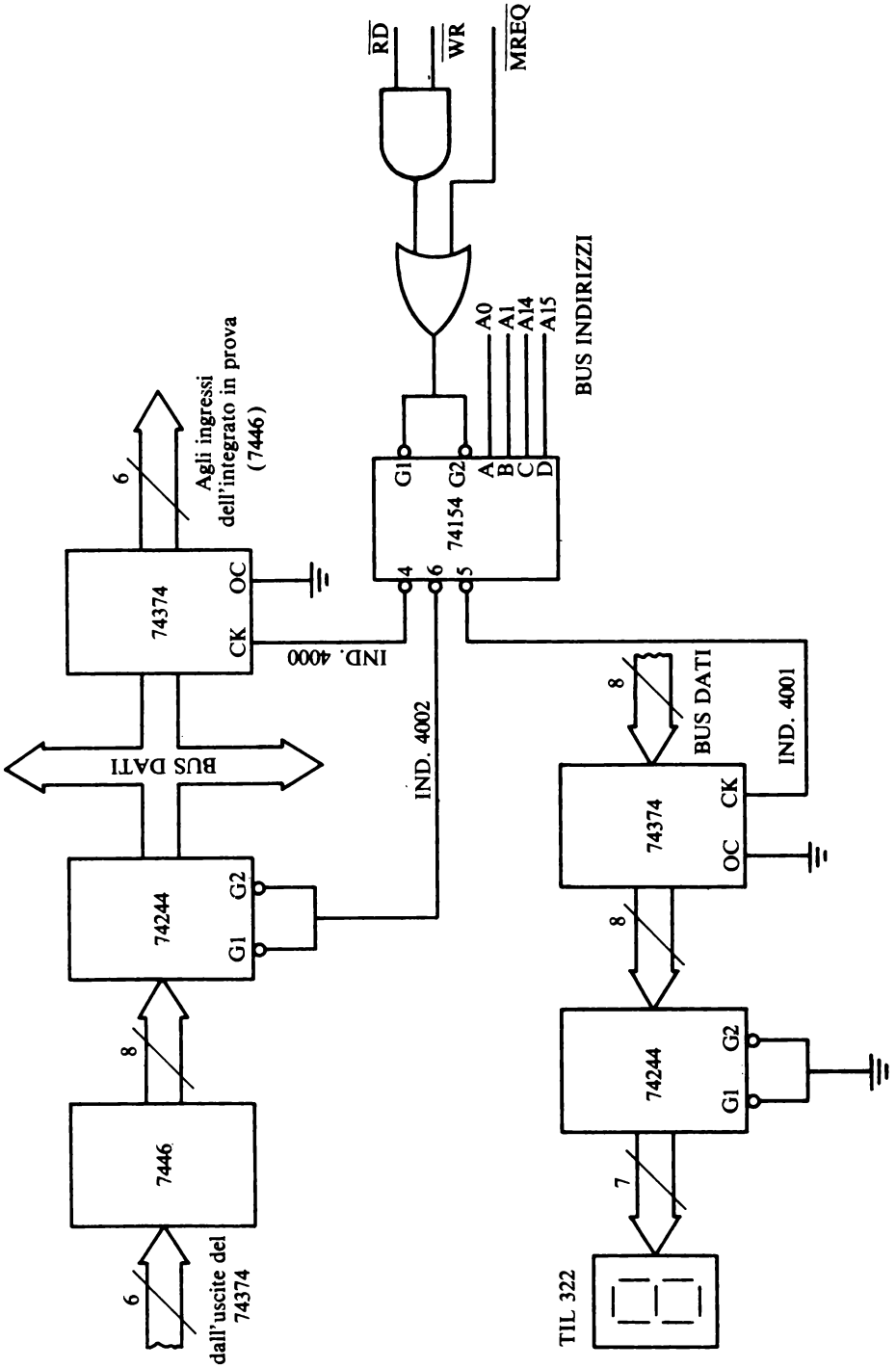


FIG. 3 - Hardware per il collaudo dell'SN 7446.

driver per il display a sette segmenti, un SN 74154 per la selezione dei porti di I/O, l'integrato SN 7446 sotto test, e due porte logiche utilizzate per l'abilitazione del demultiplexer la cui tabella funzionale è riportata in fig. 2.

La tecnica di indirizzamento dei porti di I/O è quella del «memory-mapped», realizzata utilizzando quattro linee del bus indirizzi (cap. 4°).

Il funzionamento del circuito di fig. 3 è il seguente:
mediante software vengono inviate le configurazioni di ingresso della tabella funzionale del 7446 agli ingressi RBI, LT, D, C, B, A (con RBI variabile binaria a maggior peso ed A variabile binaria a minor peso) dell'integrato sotto test attraverso il porto di uscita i cui ingressi sono collegati al bus dati e le uscite agli ingressi dello stesso 7446. Vengono quindi lette le uscite del 7446 sul bus dati attraverso il porto di ingresso 74244 e controllate con quelle scritte in tabella.

Se l'integrato testato risulta efficiente sul display verrà visualizzata la lettera B in caso contrario la lettera E.

Il flow-chart di fig. 4 e la relativa codifica del programma riportati più avanti chiariranno meglio quanto appena detto.

Naturalmente gli indirizzi assegnati ai porti di I/O non possono essere utilizzati come locazioni di memoria di lavoro e inoltre occorrerà caricare preventivamente la tabella delle configurazioni di uscita indicata in questo esempio con TAB.

ASSEMBLY

LD SP, 0800	Definisci l'area di stack.
LD A,30	Carica l'accumulatore con la prima configurazione d'ingresso (riga 0) in cui i due bit più significativi sono mascherati a 0 (00110000).
LD (4000),A	Carica il contenuto dell'accumulatore nel porto di uscita 4000; durante queste fase $\overline{\text{MREQ}}$ e $\overline{\text{WR}}$ (fig. 7) si portano al livello logico basso, il 74154 si abilita e poiché sul bus degli indirizzi viene posto 0100 0000 0000 0000 = $(4000)_{16}$, $A_0 = A = 0$, $A_1 = B = 0$, $A_{14} = C = 1$, $A_{15} = D = 0$ viene attivata l'uscita 4 del 74154 (si veda la tabella funzionale di fig. 2) ed il clock del 74374 passando dal livello logico alto a quello basso e di nuovo a quello alto porta in uscita, e quindi all'ingresso dell'integrato in prova la prima configurazione.
LD A,(4002)	Leggi il porto d'ingresso, e quindi l'uscita del 7446, e caricala in accumulatore.

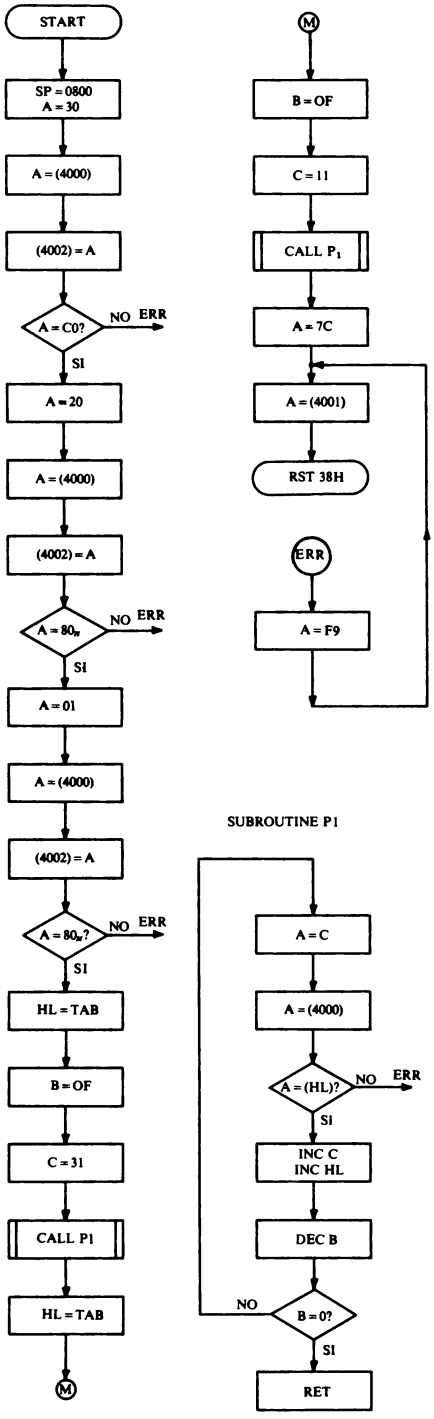


FIG. 4 - Diagramma di flusso per il collaudo dell'SN 7446.

	Durante questa istruzione $\overline{MREQ}$ e $\overline{RD}$ vengono attivati, $A_0 = 1$, $A_1 = 1$, $A_{14} = 1$, $A_{15} = 0$ e viene attivata l'uscita 6 del 74154, cosicché il dato presente all'ingresso del 74244 viene caricato in accumulatore.
CP C0	Confronta il dato caricato in accumulatore con la configurazione voluta d'uscita. Infatti C0 = 11000000 (riga 0).
JP NZ, ERR	Segnala l'errore se i due dati confrontati sono diversi.
LD A,20	
LD (4000),A	Carica il contenuto dell'accumulatore nel porto di uscita.
LD A,(4002)	Leggi il porto d'ingresso e carica il suo contenuto nell'accumulatore.
CP 80	Confrontalo con 80. In questa fase si è caricato all'ingresso del 7446 $(20)_{16}$ = 00100000 con LT = 0; di conseguenza per questa combinazione in uscita si deve ottenere 80 (ultima riga della tabella di fig. 1).
JP NZ, ERR	Segnala l'errore se il contenuto dell'accumulatore è diverso da 80. Infatti durante l'istruzione di confronto CP si altera il flag di zero a seconda del risultato, e di conseguenza il programma può saltare ad un altro punto dello stesso in funzione del risultato ottenuto.
LD A,01	Cambia la configurazione d'ingresso lasciando basso LT; si deve ottenere ancora 80.
LD (4000),A	Come sopra. Fino a questo punto del programma si è esaminata soltanto la riga 0 ed il comando LT.
LD A, (4002)	
CP 80	
JP NZ,ERR	
LD HL,TAB	Carica in HL l'indirizzo della tabella.
LD B,0F	Carica le restanti quindici combinazioni.
LD C,31	Esegui una prova con RBI alto partendo dalla configurazione 00110001 = $(31)_{16}$. I bit 7 ed 8 sono mascherati a zero.
CALL P_1	Chiama la subroutine P_1.
LD HL,TAB	Carica l'indirizzo della tabella in HL.
LD B,0F	Carica le 15 configurazioni.

	LD C,11	Esegui la prova con RBI = 0 partendo dalla configurazione 00 010001 = $(11)_{16}$ fino ad arrivare a 1F.
	CALL P_1	
	LD A,7C	Carica la configurazione 7C nell'accumulatore (g,f, e,d,c = 1; b,a = 0).
P_2	LD (4001),A	Mettila all'ingresso del driver 74244 per mezzo del porto di uscita ad esso collegato ed indirizzato con 4001 (si ripetono le stesse considerazioni fatte precedentemente per il porto di uscita 4000). Se nell'accumulatore è caricato 7C in uscita del driver si ha: a = b = 0, c = d = e = f = g = 1 ed essendo il display un catodo comune si accende la lettera b. Nel caso in cui sia caricato nell'accumulatore l'esadecimale F9 si accende la lettera E.
	RST 38H	Ridai il controllo al sistema operativo.
ERR	LDA,F9	
	JP P_2	Visualizza la lettera E (ERRORE).

Subroutine P_1

P_1	LD A,C	Carica nell'accumulatore la configurazione d'ingresso (riga 1).
	LD (4000), A	Mettila nel porto di uscita e quindi all'ingresso del 7446.
	LD A, (4002) CP (HL)	Leggi l'uscita corrispondente del 7446 e confrontala con quella memorizzata nella locazione di memoria puntata da HL.
	JP NZ, ERR	Segnala l'errore se le configurazioni sono diverse.
	INC C INC HL DJNZ P_1	Ripeti le stesse operazioni per tutte le altre configurazioni d'ingresso.
	RET	Ritorna al programma principale.

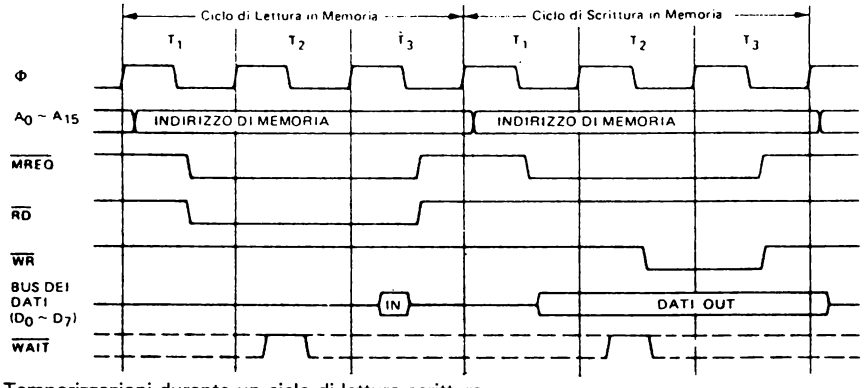


FIG. 7 - Temporizzazioni durante un ciclo di lettura-scrittura.

2. Test di collaudo del circuito integrato SN 7442

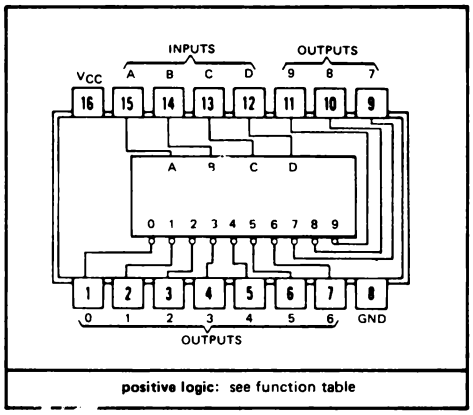
L'integrato SN 7442 (Texas) è un decodificatore BCD-Decimale la cui tabella funzionale e piedinatura sono mostrate in fig. 8.

'42A, 'L42, 'LS42 . . . BCD-TO-DECIMAL
'43A, 'L43 . . . EXCESS-3-TO-DECIMAL
'44A, 'L44 . . . EXCESS-3-GRAY-TO-DECIMAL

- **All Outputs Are High for Invalid Input Conditions**
- **Also for Application as**
  4-Line-to-16-Line Decoders
  3-Line-to-8-Line Decoders
- **Diode-Clamped Inputs**

TYPES	TYPICAL POWER DISSIPATION	TYPICAL PROPAGATION DELAYS
'42A, '43A, '44A	140 mW	17 ns
'L42, 'L43, 'L44	70 mW	49 ns
'LS42	35 mW	17 ns

SN5442A THRU SN5444A, SN54LS42 . . . J OR W PACKAGE
SN54L42 THRU SN54L44 . . . J PACKAGE
SN7442A THRU SN7444A,
SN74L42 THRU SN74L44, SN74LS42 . . . J OR N PACKAGE
(TOP VIEW)



positive logic: see function table

description

These monolithic decimal decoders consist of eight inverters and ten four-input NAND gates. The inverters are connected in pairs to make BCD input data available for decoding by the NAND gates. Full decoding of valid input logic ensures that all outputs remain off for all invalid input conditions.

The '42A, 'L42, and 'LS42 BCD-to-decimal decoders, the '43A and 'L43 excess-3-to-decimal decoders, and the '44A and 'L44 excess-3-gray-to-decimal decoders feature inputs and outputs that are compatible for use with most TTL and other saturated low-level logic circuits. D-c noise margins are typically one volt.

Series 54, 54L, and 54LS circuits are characterized for operation over the full military temperature range of −55°C to 125°C; Series 74, 74L, and 74LS circuits are characterized for operation from 0°C to 70°C.

FUNCTION TABLE

NO.	'42A, 'L42, 'LS42 BCD INPUT				'43A, 'L43 EXCESS-3-INPUT				'44A, 'L44 EXCESS-3-GRAY INPUT				ALL TYPES DECIMAL OUTPUT									
	D	C	B	A	D	C	B	A	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	L	L	L	L	L	L	H	H	L	L	H	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	L	H	L	L	L	H	H	L	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	L	H	L	H	L	H	H	H	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	L	H	H	L	L	H	L	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	L	H	H	H	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	L	L	L	H	H	L	L	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	L	L	H	H	H	L	H	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	L	H	L	H	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	L
INVALID	H	L	H	L	H	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
	H	L	H	H	H	H	H	L	H	L	L	H	H	H	H	H	H	H	H	H	H	H
	H	H	L	L	H	H	H	H	H	L	L	L	H	H	H	H	H	H	H	H	H	H
	H	H	L	H	L	L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H
	H	H	H	L	L	L	L	H	L	L	L	H	H	H	H	H	H	H	H	H	H	H
	H	H	H	H	L	L	H	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H

H = high level, L = low level

FIG. 8 - Piedinatura e tabella funzionale dell'SN 7442.

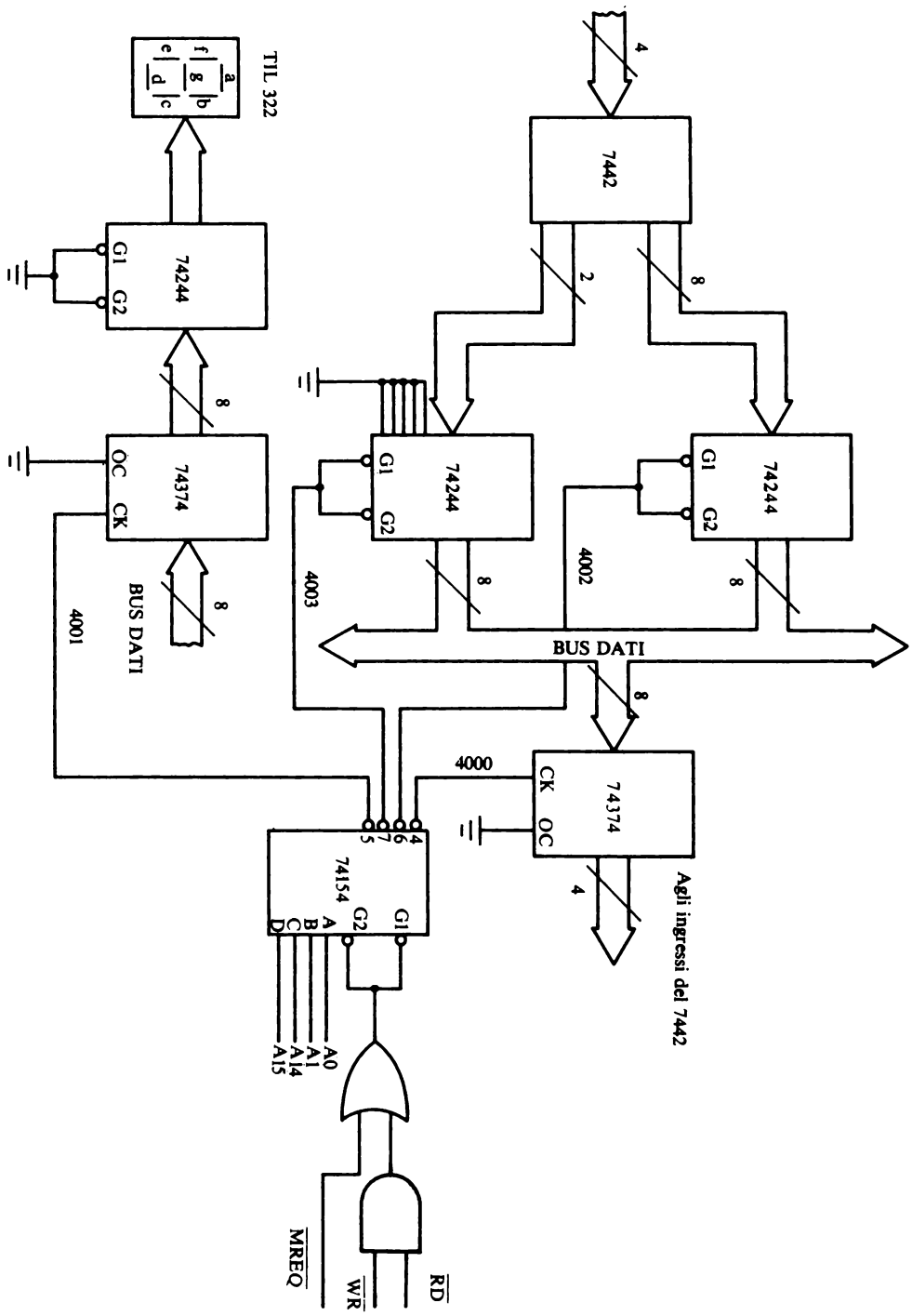


FIG. 9 - Hardware per il collaudo dell'integrato SN 7442.

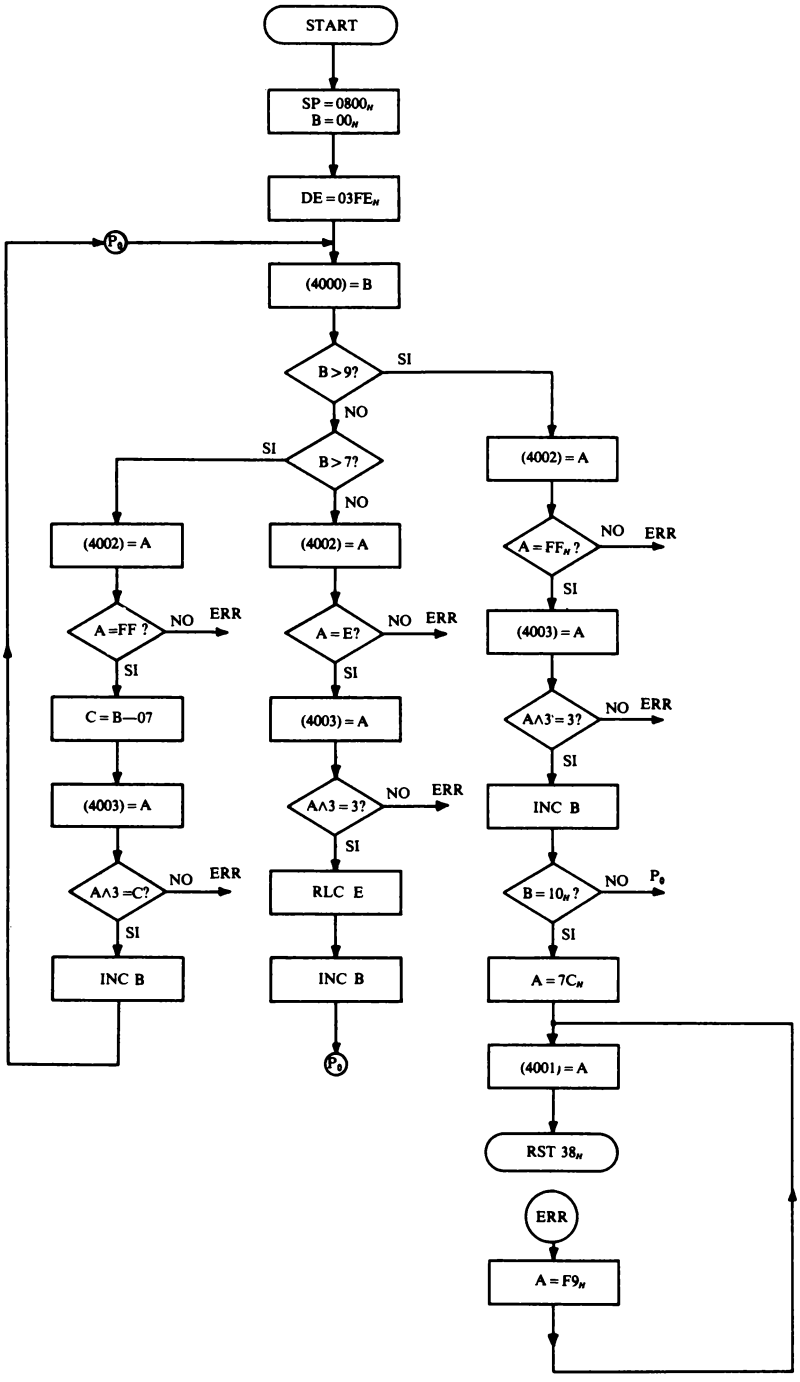


Fig. 10 - Diagramma di flusso per il collaudo dell'SN 7442.

Il software e l'hardware per il collaudo di questo integrato è leggermente diverso rispetto a quello utilizzato per il 7446 in quanto la tabella funzionale del 7442 possiede quattro ingressi e dieci uscite. Sono così necessari due porti di Input da collegare alle uscite dell'integrato sotto test come mostrato in fig. 9. L'hardware restante è del tutto identico a quello utilizzato per il 7446, così come è uguale concettualmente il funzionamento.

Il software prevede invece la lettura della tabella funzionale facendo particolare riferimento alle dieci uscite dell'integrato che si comportano allo stesso modo nelle ultime sei righe.

Il flow-chart e la susseguente codifica del programma chiariranno il procedimento seguito tenendo presente che quest'ultimo non vuole essere nè l'unico nè il più semplice ma solo uno dei possibili.

ASSEMBLY

	LD SP, 0800	Definisci l'area di stack.
	LD B,00	Azzera il contatore di configurazioni.
	LD DE,03FE	Carica nella coppia di registri DE la prima configurazione di uscita dell'integrato in prova mascherando a 0 gli ingressi, non utilizzati, del secondo porto di input a maggior peso.
P_0	LDA,B	Azzera l'accumulatore.
	LD (4000),A	Metti la prima configurazione d'ingresso (A = B = C = D = 0) sul porto di uscita e quindi all'ingresso dell'integrato in prova.
	CP 0A	Confronta il contenuto dell'accumulatore con $00001010 = (10)_{10}$. Questa istruzione in pratica verifica se si sono esaminate le prime dieci righe della tabella funzionale; con questa istruzione vengono modificati i flag di carry e di zero a seconda del risultato.
	JP NC, P_1	Salta a P_1 se non c'è riporto (se B>9).
	CP 08 JP NC,P_2	Confronta il contenuto dell'accumulatore con 08, se il contenuto dell'accumulatore e quindi del registro contatore B, è maggiore di 07 salta a P_2.
	LD A,(4002)	Leggi il primo porto d'ingresso e metti il suo contenuto in accumulatore.

	CP E	Confronta il contenuto dell'accumulatore con quello del registro E, alla prima lettura E contiene FE.
	JP NZ,ERR	Segnala l'errore se i contenuti sono diversi.
	LD A,(4003)	Leggi il secondo porto di input.
	AND 03	Esegui il prodotto logico di 03 con il contenuto dell'accumulatore.
	CP 03	Confronta il contenuto dell'accumulatore con 03.
	JP NZ,ERR	Segnala l'errore se i due numeri sono diversi.
	RLC E INC B	Leggi la successiva configurazione della tabella del 7442. Se, per esempio, il contenuto del registro E era uguale ad FE dopo queste istruzioni il suo contenuto vale FD e il flag di carry è messo ad 1.
	JP P_0	Salta a P_0.
P_1	LDA, (4002)	Leggi il primo porto di input e metti il suo contenuto in accumulatore.
	CP FF	Confronta il contenuto dell'accumulatore con FF.
	LD A,(4003)	Leggi il secondo porto di input.
	AND 03	Maschera i sei bit più significativi dell'accumulatore.
	CP 03	Confronta il valore così ottenuto con 03.
	JP NZ, ERR	Segnala l'errore se i due numeri sono diversi.
	INC B	Incrementa il contatore di configurazioni.
	LD A,10	Carica nell'accumulatore $(16)_{10}$, cioè il numero delle righe da leggere.
	CP B	Confronta il contenuto dell'accumulatore con il contatore di configurazioni.
	JP NZ,P_0	Salta a P_0 se non sono state lette tutte le configurazioni.
	LDA,7C	Visualizza la lettera b se il test è positivo.
P_3	LD(4001),A	
	RST 38H	Ridai il controllo al sistema operativo.
ERR	LDA,F9 JP P_3	Visualizza la lettera E se il test è negativo.
P_2	LDA,(4002)	Leggi il primo porto di input e poni il suo contenuto in accumulatore.
	CP FF	Confrontalo con FF.
	JP NZ,ERR	Segnala l'errore se i due numeri sono diversi.

LD A,B
SUB 07
LD C,A
	Costruiscono i due bit di confronto con cui comparare quelli successivamente letti dal porto (4003). Essendo questi due bit 01 e 10 rispettivamente per l'ottava e nona configurazione di prova, è sufficiente una semplice sottrazione in riferimento a B.

LD A,(4003)	Leggi il secondo porto di input e metti il risultato in accumulatore.
AND 03	Esegui il prodotto logico tra 03 e il contenuto dell'accumulatore, se il risultato è 03 i due contenuti sono uguali ed è messo ad 1 il flag di zero altrimenti è messo a zero.
CP C	Confronta il contenuto dell'accumulatore con 01. Se i due numeri sono uguali il flag di zero è messo ad 1.
JP NZ,ERR	Segnala l'errore se i dati confrontati sono diversi, cioè salta ad ERR se il risultato del confronto è diverso da zero ovvero il flag di zero è uguale ad 1.
INC B	Incrementa il contatore di configurazioni.
JP P_0	Salta a P_0.

3. Semplice applicazione di un sistema a microprocessore

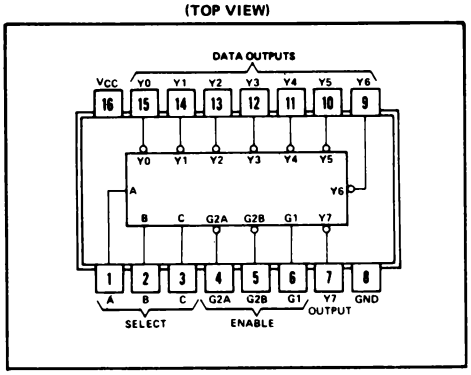
Analisi Hardware

Lo schema di fig. 12 illustra un sistema minimo programmabile, a microprocessore.

Tale sistema, basato sul μP Z80, è fornito di 2 K byte residenti in EPROM e di 2 K byte di memoria RAM. Dato il sistema di indirizzamento ambiguo (non vengono utilizzati i 4 bit più significativi dell'indirizzo) i due chip di memoria si vedono allocati in ciascuno dei 16 gruppi di 4 K byte che la CPU può indirizzare direttamente, nei quali la EPROM risponde ai primi 2048 indirizzi di ogni gruppo mentre la RAM ai rimanenti 2048. Le uscite della EPROM sono abilitate solo in fase di lettura; questo è fatto per evitare conflitti di correnti sul bus dati durante una accidentale scrittura in EPROM. La CPU è pilotata da un circuito quarzato oscillante a 4 MHz che, tramite un flip-flop montato in configurazione T, le fornisce un clock di sistema a 2 MHz. Il dialogo con il mondo esterno, che nel caso specifico si identifica con l'operatore, è reso possibile da due porti di I/0 indirizzati con la tecnica dell'I/0 isolato. Il primo porto, utilizzato solo per l'ingresso dei dati, risponde ai codici dispositivo (41), (49), (C1), e (C9) in quanto anche qui si è in presenza di un indirizzamento ambiguo, ed è costituito da un buffer trigger di Schmitt tristate ottuplo che, quando è letto, pone sul bus dei dati l'informazione binaria impostata sugli otto

switch (aperto = 0, chiuso = 1). Il secondo porto è utilizzato unicamente in uscita dal sistema, risponde ai codici dispositivo (40), (C0), (48) e (C8), ed è costituito da un buffer latch ottuplo che quando è selezionato pone permanentemente in uscita su otto led il byte appena inviatogli dalla CPU. Qualsiasi tentativo della CPU di leggere dal porto di uscita o di scrivere nel porto di ingresso non produce nessun effetto in quanto i segnali di abilitazione dei porti (codice dispositivo + $\overline{IORQ}$) vengono filtrati, tramite porte logiche Or, dai segnali $\overline{RD}$ e $\overline{WR}$. Il sistema è inoltre dotato di circuiti per la generazione normale, ed automatica all'atto dell'accensione, di impulsi di reset e di richieste di interruzioni mascherabili. All'accensione l'impulso $\overline{INT}$ non ha alcun effetto poiché il $\overline{RESET}$ è prioritario e dura più dell'impulso $\overline{INT}$.

SN54LS138, SN54S138 . . . J OR W PACKAGE
SN74LS138, SN74S138 . . . J OR N PACKAGE
(TOP VIEW)



'LS138, 'S138
FUNCTION TABLE

INPUTS					OUTPUTS							
ENABLE		SELECT										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

*G2 = G2A + G2B
H = high level, L = low level, X = irrelevant

FIG. 11 - Piedinatura e tabella della verità dell'SN 74LS138.

Analisi Software

Anzitutto il programma ha come scopo quello di illustrare l'utilizzo dei due porti di I/0 per comunicazioni (sistema) ← → (mondo esterno). Il programma verte su un loop di interrogazione del porto di ingresso dal quale, se risulta azionato un tasto, si passa ad un ramo che determina quale tasto è stato premuto ed esegue un sottoramo opportuno. Innanzitutto tutto il programma esegue delle inizializzazioni: viene definita l'area di stack, visto che si fa uso di una subroutine, caricando il registro SP con l'ultimo indirizzo di RAM disponibile (0FFF) poiché lo stack cresce verso il basso; viene inizializzato a 01 il registro C, mentre viene azzerato il registro B utilizzato come contatore.

Di qui si entra nel loop di interrogazioni con l'acquisizione nell'accumulatore dello stato attuale degli otto switch. Se tutti gli interruttori sono

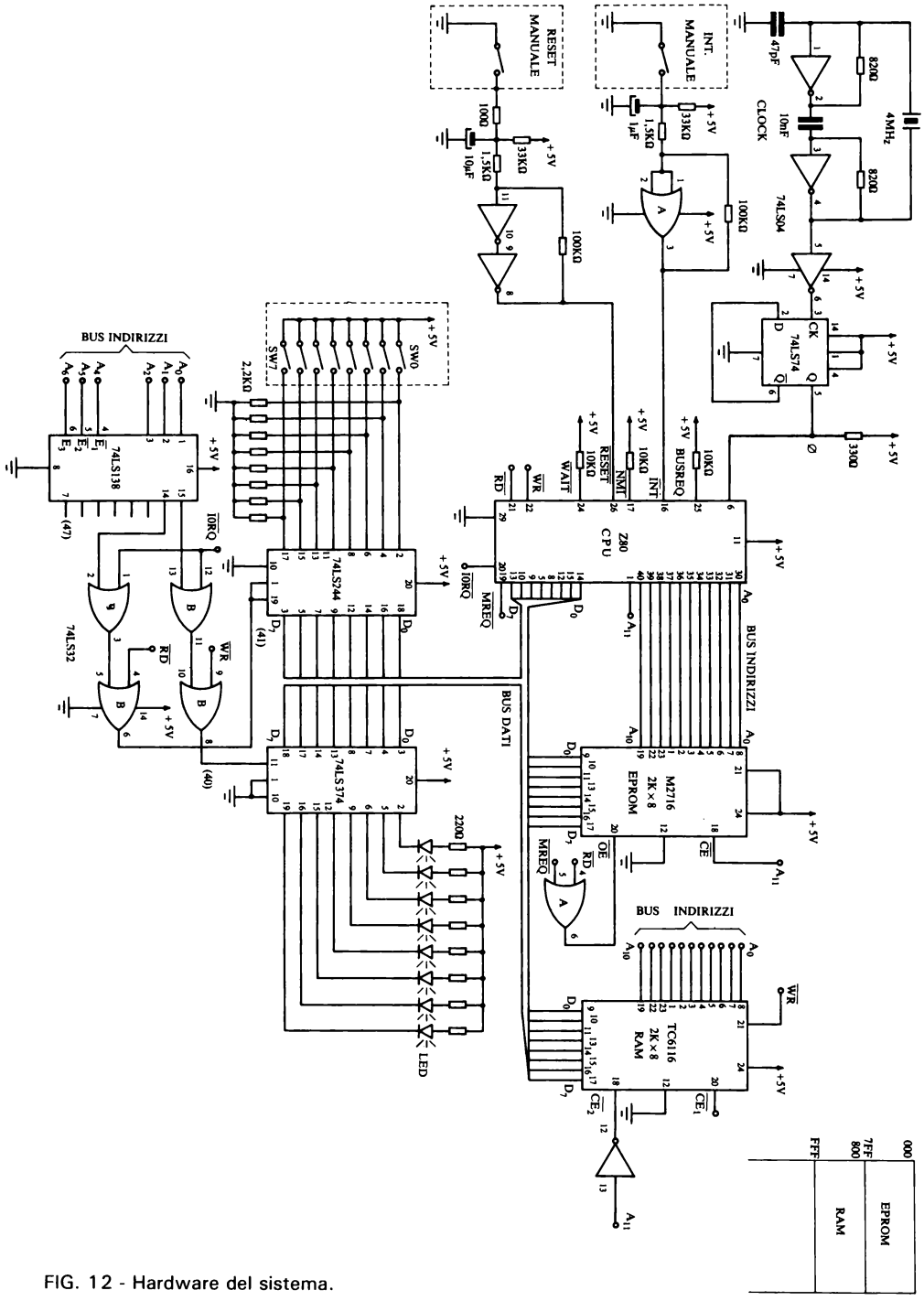


FIG. 12 - Hardware del sistema.

inattivi, tutti i bit sono uguali a zero, quindi l'operatore non ha richiesto alcunché, i registri C e B vengono reinizializzati ed i led vengono spenti. Da notare che per ricopiare sui led il contenuto dell'accumulatore è necessario complementare tutti i bit mediante l'istruzione CPL in quanto, come si nota dallo schema hardware uno 0 logico in uscita accende il led anziché spegnerlo.

Se invece l'accumulatore contiene un byte diverso da 0 vuol dire che è stato premuto almeno un tasto e si passa tramite una serie di istruzioni BIT n, A a testare i bit dell'accumulatore nell'ordine $b_0 \rightarrow b_7$; non appena viene individuato il primo bit a 1 si esegue il ramo associato a quel bit, cioè a quel tasto.

Se viene premuto il *tasto 0* viene caricato in accumulatore il contenuto del contatore B, quindi viene eseguita la subroutine che, come si vedrà, esegue l'output e genera un ritardo; quindi viene incrementato il contatore B e si ritorna al loop di interrogazione. Da qui, se il tasto è ancora premuto, si ritorna al ramo precedente e B viene nuovamente copiato sui led ed incrementato, e così via. Una cosa analoga accade per i *tasti 1 e 2* con la differenza che invece del contatore B che si incrementa ad ogni ciclo, viene visualizzato il registro C il cui bit a 1 viene fatto ruotare circolarmente a sinistra e a destra rispettivamente.

Per il *tasto 3* viene visualizzata per ogni ciclo di interrogazione prima la configurazione 00001111 e poi la 11110000.

Per il *tasto 4* invece ad ogni ciclo di interrogazione viene visualizzata prima la configurazione 10101010 e successivamente la 01010101.

I rami fin qui analizzati fanno uso della subroutine DELAY.

Tale routine si occupa prima di tutto della trasmissione del byte, previa complementazione per il motivo già detto. Fatto ciò genera un ritardo, di durata costante, mediante l'utilizzo di due loop nidificati; tale ritardo è di circa 0,45 secondi per una frequenza di clock di 2 MHz. Ciò vuol dire che dopo ogni OUT il programma si ferma per poco meno di mezzo secondo, tranne nel caso in cui nessun tasto sia stato premuto e nel caso sia attivo il tasto 5,6, o 7. In questi casi infatti ad ogni rispettiva istruzione di OUT, eseguita in proprio, fa immediatamente seguito il ritorno al loop di interrogazione.

Per quanto riguarda i rami relativi ai *tasti 5,6 e 7*, data la mancanza dell'istruzione CPL, essi emetteranno anziché i valori esadecimali 20,40 e 80, i loro complementi, cioè accenderanno tutti i led tranne il 6°, il 7° e l'8° rispettivamente.

Da notare, data la scansione sequenziale dei bit letti dal porto di ingresso, il primo tasto attivo meno significativo sarà prioritario ed escluderà eventuali altri.

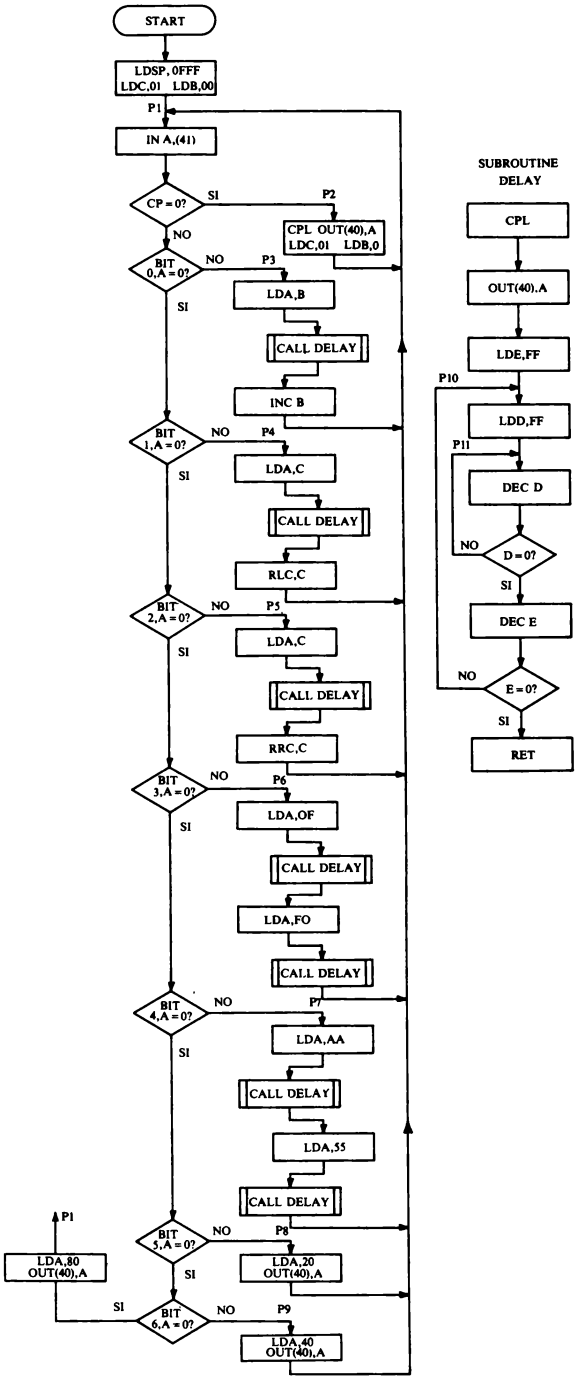


FIG. 13 - Diagramma di flusso.

Conclusioni

All'accensione il sistema va in reset automaticamente e, **presupponendo il programma caricato in EPROM,** la CPU esegue subito il programma e comincia a leggere lo stato dei tasti. Il comportamento dei led è il seguente (fig. 14):

Tasto premuto	Alla pressione del tasto	Al rilascio del tasto
Nessuno	Tutti i LED sono spenti.	——
0	I LED contano in binario alla cadenza di circa 0,5 secondi. Il conteggio riprende dal valore con cui si era interrotto precedentemente a meno che nel frattempo non sia stata letta almeno una volta la configurazione 00 nel qual caso il contatore è resettato.	La funzione si interrompe dopo circa 0,5 sec. dall'ultima transizione.
1	Un LED acceso ruota circolarmente verso sinistra alla cadenza di circa 0,5 sec. Stesse considerazioni sulla posizione inziale del led, che per il tasto numero zero.	Come al tasto 0.
2	Un LED acceso ruota circolarmente verso destra alla cadenza di circa 0,5 sec. Stesse considerazioni sulla posizione iniziale del led, che per il tasto numero zero.	Come al tasto 0.
3	Appare la configurazione 00001111 seguita dalla 11110000, ciascuna per circa 0,5 sec.	La funzione si interrompe dopo circa 0,5 secondi dall'apparizione della configurazione 11110000.
4	Appare la configurazione 10101010 seguita dalla 01010101, ciascuna per circa 0,5 sec.	La funzione si interrompe dopo circa 0,5 secondi dall'apparizione della configurazione 01010101.
5	Appare la configurazione 11011111	La funzione si interrompe immediatamente.
6	Appare la configurazione 10111111	Come al tasto 5.
7	Appare la configurazione 01111111	Come al tasto 5.

Note: Nelle configurazioni: 0 = LED spento, 1 = LED acceso.

Al termine di ogni funzione diventa attiva la funzione abbinata al primo tasto premuto meno significativo.

FIG. 14

```
                                                                    MICRO     Page    1
ln  addr  obj code t e i line   source statement                     SGS asz80 Rel. 1.0

 1                        1          .ORG    0000H
 2  0000  31FF0F          2          LD      SP,0FFFH
 3  0003  0E01            3          LD      C,01H          ; C=00000001
 4  0005  0600            4          LD      B,00H          ; INIZ. CONT. B
 5  0007  DB41            5  P1:     IN      A,(41H)        ; LEGGI SWITCH
 6  0009  FE00            6          CP      00H            ; PREM. TASTI?
 7  000B  CA3B00          7          JP      Z,P2
 8  000E  CB47            8          BIT     0,A            ; PREM. TASTO 0?
 9  0010  C24200          9          JP      NZ,P3          ; ESEGUI RAMO
10  0013  CB4F           10          BIT     1,A            ;    "      "     1
11  0015  C24A00         11          JP      NZ,P4          ;    "      "
12  0018  CB57           12          BIT     2,A            ;    "      "     2
13  001A  C25300         13          JP      NZ,P5          ;    "      "
14  001D  CB5F           14          BIT     3,A            ;    "      "     3
15  001F  C25C00         15          JP      NZ,P6          ;    "      "
16  0022  CB67           16          BIT     4,A            ;    "      "     4
17  0024  C26900         17          JP      NZ,P7          ;    "      "
18  0027  CB6F           18          BIT     5,A            ;    "      "     5
19  0029  C27600         19          JP      NZ,P8          ;    "      "
20  002C  CB77           20          BIT     6,A            ;    "      "     6
21  002E  C27D00         21          JP      NZ,P9          ;    "      "
22  0031  3E80           22          LD      A,80H          ; LED=01111111
23  0033  D340           23          OUT     (40H),A        ; OUT VAL. 80H
24  0035  C30700         24          JP      P1             ;
25  0038  2F             25  P2:     CPL                    ; COMPLEMENTA A
26  0039  D340           26          OUT     (40H),A        ; A=FFH LED OFF
27  003B  0E01           27          LD      C,01H
28  003D  0600           28          LD      B,00H
29  003F  C30700         29          JP      P1             ; RIENTRA
30  0042  78             30  P3:     LD      A,B
31  0043  CD8400         31          CALL    DELAY          ; OUT DI B
32  0046  04             32          INC     B
33  0047  C30700         33          JP      P1             ; RIENTRA
34  004A  79             34  P4:     LD      A,C
35  004B  CD8400         35          CALL    DELAY          ; OUT DI C
36  004E  CB01           36          RLC     C              ; AGGIORNA C
37  0050  C30700         37          JP      P1             ; RIENTRA
38  0053  79             38  P5:     LD      A,C
39  0054  CD8400         39          CALL    DELAY          ; OUT DI C
40  0057  CB09           40          RRC     C              ; AGGIORNA C
41  0059  C30700         41          JP      P1             ; RIENTRA
42  005C  3E0F           42  P6:     LD      A,0FH
43  005E  CD8400         43          CALL    DELAY          ; OUT 00001111
44  0061  3EF0           44          LD      A,0F0H
45  0063  CD8400         45          CALL    DELAY          ; OUT 11110000
46  0066  C30700         46          JP      P1             ; RIENTRA
47  0069  3EAA           47  P7:     LD      A,0AAH
48  006B  CD8400         48          CALL    DELAY          ; OUT 10101010
49  006E  3E55           49          LD      A,055H
50  0070  CD8400         50          CALL    DELAY          ; OUT 01010101
51  0073  C30700         51          JP      P1             ; RIENTRA
52  0076  3E20           52  P8:     LD      A,020H         ; LED=11011111
53  0078  D340           53          OUT     (40H),A        ; OUT VAL. 20H
54  007A  C30700         54          JP      P1             ; RIENTRA
55  007D  3E40           55  P9:     LD      A,40H          ; LED=10111111
56  007F  D340           56          OUT     (40H),A        ; OUT VAL. 40H
57  0081  C30700         57          JP      P1             ; RIENTRA


                                                                    MICRO     Page    2
ln  addr  obj code t e i line   source statement                     SGS asz80 Rel. 1.0

 1  0084  2F             58  DELAY:  CPL                    ; COMPLEMENTA A
 2  0085  D340           59          OUT     (40H),A        ; OUT DI A
 3  0087  1EFF           60          LD      E,0FFH         ; INIZ. I CONT.
 4  0089  16FF           61  P10:    LD      D,0FFH         ;    "      "
 5  008B  15             62  P11:    DEC     D              ; LOOP INTERNO
 6  008C  C28B00         63          JP      NZ,P11
 7  008F  1D             64          DEC     E              ; LOOP ESTERNO
 8  0090  C28900         65          JP      NZ,P10
 9  0093  C9             66          RET                    ; RITOR. PROG.
10                       67          .END
    0 error(s)
```

4. Semplice videogioco realizzato a microprocessore

Analisi Hardware

Lo schema di figura 15 illustra un sistema a microprocessore che contiene tutti i componenti essenziali per essere definito microcomputer. In esso infatti si distinguono una CPU, un generatore di clock, una RAM, una ROM, dei semplici porti di I/0 ed un dispositivo di interfaccia parallela PIO. Il μprocessore è uno Z80-A in tutto e per tutto simile allo Z80 rispetto al quale però ha il vantaggio di poter sopportare frequenze di lavoro fino a 4MHz. Il clock di sistema infatti ha proprio tale frequenza ed è fornito, alla CPU ed al dispositivo PIO, da un flip-flop collegato in configurazione T, pilotato a sua volta da un segnale a 8 MHz generato da un circuito quarzato.

Il μcomputer è dotato di 2 Kbyte di memoria residente in EPROM e da 2 Kbyte di memoria RAM. Ad ogni dispositivo di memoria fanno quindi capo 11 linee di indirizzamento (le meno significative $A_0 \div A_{10}$). La linea di indirizzamento A_{11} provvede a selezionare l'opportuno integrato di memoria cosicché i 4 Kbyte complessivi di memoria sono allocati in maniera contigua l'uno rispetto all'altro. Più precisamente, non essendo utilizzate le 4 linee di indirizzamento più significative ($A_{12} \div A_{15}$), i 4Kbyte di memoria fisica del μcomputer si trovano praticamente allocati in ognuno dei 16 banchi logici da 4Kword che il μPZ80-A può gestire linearmente. In ognuno di questi 16 banchi la EPROM risponde ai primi 2048 indirizzi (X000 ÷ X7FF) mentre la RAM ai rimanenti 2048 (X800 ÷ XFFF). L'abilitazione del dispositivo RAM è affidata alle linee $\overline{MREQ}$ e A_{11}, mentre quella del dispositivo EPROM alle linee A_{11} e $(\overline{RD} + \overline{MREQ})$. Quest'ultima linea previene l'abilitazione del driver di uscita della EPROM in presenza di una illecita operazione di scrittura nella EPROM stessa; ciò al fine di evitare conflitti di tensione e/o correnti sul bus dati che potrebbero danneggiare i dispositivi ad esso collegati.

Per quanto riguarda la circuiteria di I/0 essa è formata da un porto di ingresso a 8 bit, uno di uscita sempre a 8 bit, un dispositivo PIO che può fornire due porte di I/0 programmabili e dalla rete logica per la generazione di impulsi di selezione dispositivo, dato che essi sono indirizzati con la tecnica dell'I/0 isolato.

Il porto di ingresso è costituito da un buffer trigger di Schmitt three-state ottuplo (74244) il quale, quando è abilitato, trasferisce sul bus dati lo stato degli interruttori collegati ai suoi ingressi (interruttore aperto = 0 logico, interruttore chiuso = 1 logico). Quando non è abilitato le sue uscite sul bus dati sono in alta impedenza. Il secondo porto invece è utilizzato unicamente per l'uscita ed è costituito da un buffer latch ottuplo con uscite three-state (che però in questo caso sono permanentemente abilitate). Quando ricevono un impulso di abilitazione gli otto flip-flop acquisiscono il byte presente sul bus dati e lo pongono permanentemente sulle loro usci-

te le quali pilotano direttamente i catodi di 8 diodi LED. Lo 0 logico accenderà quindi il LED mentre l'1 logico lo terrà spento.

La generazione degli impulsi di selezione dispostivo è affidata ad un demultiplexer 3→8 e ad un semplice circuito combinatorio. In particolare il demultiplexer (74138) esegue la decodifica della parte bassa del bus indirizzi interpretandola come codice dispositivo; il suo segnale di selezione dispositivo, attivo basso, è poi convalidato dal segnale $\overline{IORQ}$ tramite un operatore OR in modo tale che raggiunga il dispositivo (porti di I/0) solamente durante delle regolari operazioni di I/0. Inoltre qualsiasi tentativo da parte della CPU di leggere dal porto di uscita, o di scrivere nel porto di ingresso, non ha alcun effetto in quanto i segnali di abilitazione dei porti ($\overline{\text{codice dispositivo}} + \overline{IORQ}$) vengono ulteriormente filtrati, tramite operatori OR, dalle linee $\overline{RD}$ e $\overline{WR}$. Da notare che il demultiplexer esegue una decodifica ambigua del codice dispositivo in quanto non utilizza le linee A_3 ed A_7. Questo fa si che il porto di ingresso non sia identificabile col solo codice dispositivo 41, ma anche con 49, C1, e C9, casi in cui variano appunto i bit 3 e 7; in maniera analoga anche il porto di uscita non fa capo al solo codice 40 ma anche a 48, C0, e C8.

Lo stesso demultiplexer si incarica inoltre di generare i segnali di selezione per il dispositivo PIO. Tale dispositivo è in grado di fornire al sistema due porte di I/0 parallele a 8 bit programmabili e dotate ciascuna di linee di controllo handshake con il dispositivo periferico. Le due porte del PIO inoltre hanno la possibilità, se programmate adeguatamente, di generare richieste di interruzione alla CPU in risposta a particolari cambiamenti o comunicazioni del dispositivo periferico che stanno interfacciando. La CPU quando dialoga col dispositivo PI0 deve indirizzare univocamente in esso l'utilizzo della porta A o della porta B. Deve altresì specificare se l'operazione riguarda la programmazione della porta oppure se si tratta di una normale operazione di I/0 su quella particolare porta. A questo scopo il PIO è dotato oltre al generico pin di chip-enable (CE) di altri due ingressi, A/B e C/D, che a seconda del loro stato informano il PI0 del tipo di operazione che la CPU intende eseguire su di esso: se l'ingresso A/B è 0 l'operazione fa riferimento alla porta A (diversamente alla B), inoltre se l'ingresso C/D è 0 è selezionato il modo DATA (normali operazioni di I/0 con la porta), mentre se è 1 è selezionato il modo CONTROL (programmazione della porta).

Un sistema per fornire delle informazioni a questi due ingressi del PIO durante le operazioni di I/0 è quello di utilizzare le linee A_0 e A_1 in quanto il loro valore è direttamente stabilito da software durante un'istruzione di I/0 dal codice dispositivo scelto. Questo sistema in pratica offre lo svantaggio di dover utilizzare per il dialogo con il PIO ben 4 codici dispositivo diversi che varieranno fra loro almeno per i primi 2 bit (LSB). In pratica è come se la CPU vedesse il PIO come 4 periferici: il periferico «porta A», il periferico per la programmazione della porta A, il periferico «porta B», e il periferico per la programmazione della porta B. Come già accennato lo

stesso demultiplexer si preoccupa di fornire le decodifiche di questi 4 codici che in questo caso sono i codici 44, 45, 46 e 47. La contiguità di questi 4 valori ci assicura la variabilità dei 2 LSB mentre la configurazione dei 4 MSB (X100) permette di utilizzare lo stesso demultiplexer di prima. In condizioni di riposo le uscite del demultiplexer sono a livello alto mentre in fase di decodifica l'uscita selezionata si porta a livello basso. La rete di operatori AND ha perciò la funzione di trasmettere, su un'unica linea, uno 0 logico se questo si manifesta ad una delle 4 uscite del demultiplexer (codici dispositivo 44, 45,46 e 47). Come al solito il demux esegue una decodifica ambigua delle linee $A_0 \div A_7$, tuttavia lo 0 logico raggiunge il pin $\overline{CE}$ del PIO solo ed unicamente in presenza delle 4 configurazioni scelte in quanto tutte le altre variazioni delle linee A_3 e A_7 vengono filtrate dalle linee stesse tramite due operatori OR. Da notare che nella generazione del segnale CE non concorrono le linee $\overline{IORQ}$, $\overline{RD}$ o $\overline{WR}$ in quanto queste sono già presenti in ingresso sul PIO stesso.

Riassumendo, l'ingresso $\overline{CE}$ del PIO è attivo solamente quando sulle linee $A_0 \div A_7$ sono presenti le configurazioni $44 \div 47$. Quando il PIO è abilitato è in grado di riconoscere se si è richiesto il suo intervento testando il suo ingresso $\overline{IORQ}$. In questo caso le linee A_0 ed A_1 oltre a concorrere nella generazione del segnale $\overline{CE}$ informano pure il PIO, insieme alla linea $\overline{RD}$, su che tipo di operazione gli è richiesto di fare (lettura, scrittura, programmazione, porta A, porta B) come è qui di seguito riportato:

Codice dispositivo	$\overline{IORQ}$	$\overline{RD}$	Operazione
44	0	0	lettura byte dalla porta A
44	0	1	scrittura byte nella porta A
45	0	0	lettura byte dalla porta B
45	0	1	scrittura byte nella porta B
46	0	1	byte di programmazione per la porta A
47	0	1	byte di programmazione per la porta B
xx	1	x	non è un'operazione di I/0

Il pin IEI del PIO è posto a livello logico alto: questo informerà il PIO che all'atto di una eventuale richiesta di interruzione egli non avrà vincoli di priorità da rispettare.

I dispositivi con cui sono collegate le porte del PIO sono analoghi a quelli dei due porti di I/0 in dotazione al μcomputer: una batteria di 8 led per la porta B (out) e otto switches per la porta A (in) in cui switch aperto = 1 logico e switch chiuso = 0 logico. Per la porta A, funzionante in modo ingresso per il software in dotazione, è necessario che il dispositivo esterno (in questo caso l'operatore umano) segnali al PIO di prendere in

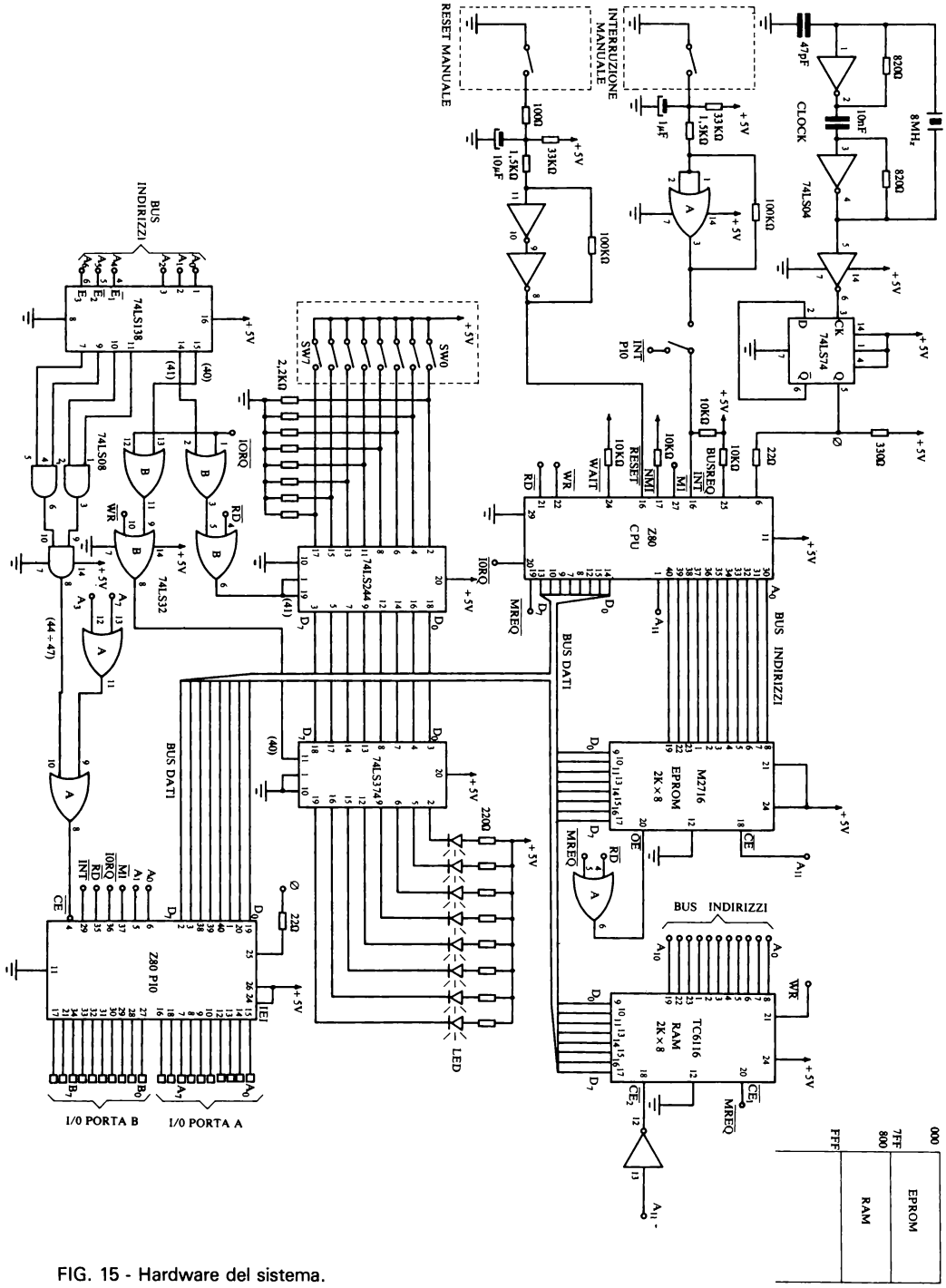


FIG. 15 - Hardware del sistema.

considerazione ed acquisire l'informazione a 8 bit presente sulle linee della porta, ed eventualmente, se abilitata, generare una richiesta di interruzione alla CPU. Tale segnalazione avviene grazie alla chiusura del tasto T (fig. 15): in queste condizioni il monostabile genera un breve impulso basso sulla linea $\overline{\text{ASTB}}$.

Il μcomputer inoltre è dotato di circuiti per la generazione manuale ed automatica all'accensione di segnali $\overline{\text{INT}}$ e $\overline{\text{RESET}}$. All'accensione tuttavia il segnale $\overline{\text{INT}}$ non ha alcun effetto in quanto, oltre ad avere una durata minore del segnale $\overline{\text{RESET}}$, quest'ultimo è prioritario e disabilita istantaneamente le interruzioni.

Analisi Software

Il software in dotazione al sistema, oltre ad essere un programma didattico, costituisce un buon esempio di programmazione con utilizzo di dispositivi PIO e relative tecniche di interruzione. Il programma è costituito principalmente da: un ramo di inizializzazione, nel quale tutto il sistema è preparato e predisposto per l'esecuzione del programma; da un breve ciclo, che in pratica costituisce il nucleo di tutto il processo; da una subroutine che genera un ritardo variabile; e da una routine di gestione dell'interruzione, eseguita ogni qualvolta l'operatore genera una segnale ASTB, mediante il tasto apposito, chiedendo che venga presa in considerazione la combinazione degli switches alla porta A.

Come si è detto il programma è in pratica un gioco, un computer game piuttosto rudimentale. Lo scopo del giocatore è quello di replicare, in tempo e col minor numero di errori possibile, una certa configurazione che il μcomputer emette e cambia dopo un certo intervallo di tempo. Egli ha la possibilità di fare due errori, dopodiché il programma lo facilita aumentando il tempo fra il cambio di una configurazione e l'altro e dandogli altre due possibilità. Dopo la 4ª facilitazione il programma si arresta. Se il giocatore replica esattamente la configurazione il gioco segnala la vittoria. In fig. 16 e 17 vengono riportati i diagrammi di flusso dell'applicazione proposta.

1. Inizializzazione

All'accensione la CPU va in reset automaticamente grazie alla rete RC apposita; **questo fa sì che inizi l'esecuzione del programma dalla locazione 0000, in EPROM, ove trova un'istruzione di salto al ramo di inizializzazione.** Tale istruzione di salto si sarebbe potuta evitare allocando il ramo di inizializzazione dalla locazione 0000 in poi, senonché tale soluzione avrebbe sconvolto l'utilizzo di una istruzione RST_{10} (o di una qualsiasi altra restart in pagina zero) che viene infatti invocata più avanti nella elabo-

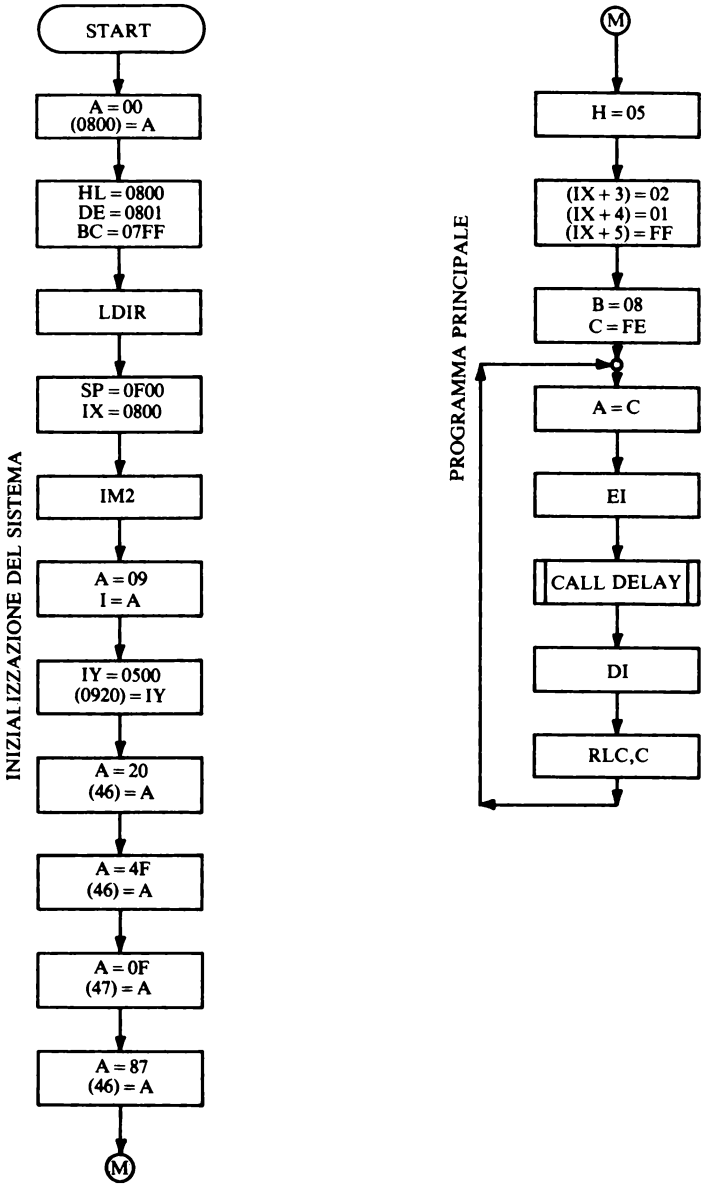
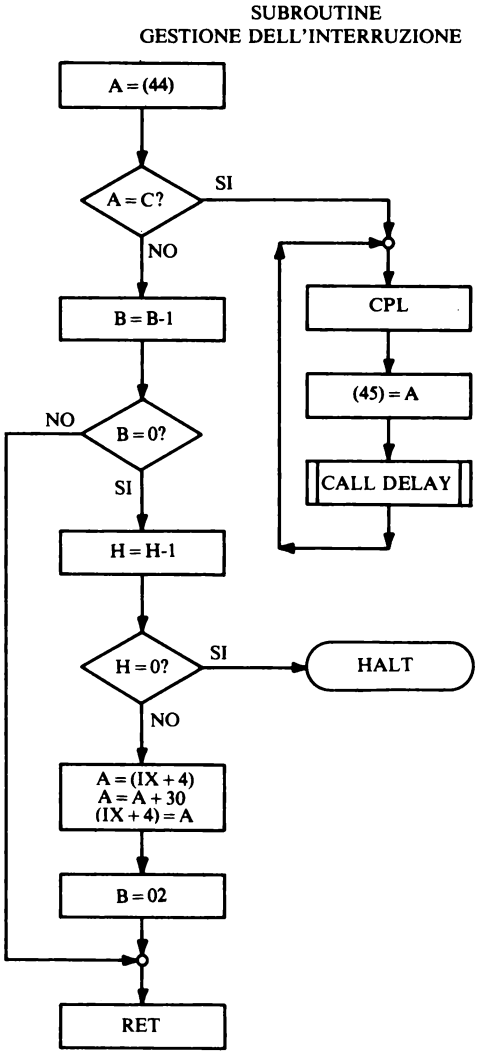


FIG. 16 - Diagramma di flusso.



Note:

— La resistenza di 22Ω, collegata al pin 6 della CPU, ha la funzione di adattare la linea di clock. Non è necessaria per linee di clock corte.

— Le resistenze collegate agli ingressi del 74LS244 assicurano il livello logico zero nelle condizioni di tasto o pulsante aperto. Il loro valore è stato scelto in modo che sia verificata la condizione

$$I_{ILMAX} \cdot R \leq V_{ILMAX}$$

(per il 74LS244 I_{ILMAX} = 0,2 mA e V_{ILMAX} = 0,8 V)

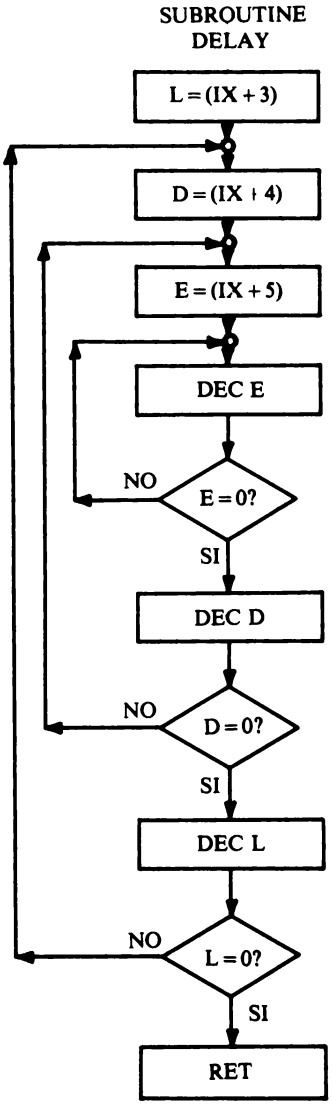


FIG. 17 - Subroutine delay.

razione portando il processo ad una istruzione di HALT che lo blocca definitivamente. Non appena il controllo viene fornito al ramo di inizializzazione la CPU, con le prime sei istruzioni, azzera tutta la memoria RAM in dotazione, azione non necessaria quanto però apprezzabile. Difatti, dopo averlo azzerato, l'accumulatore viene copiato alla locazione 800 (inizio RAM), le coppie HL e DE vengono caricate con i primi due indirizzi di RAM rispettivamente, e la coppia BC viene posta a 07FF (2047_{10}). Quindi con l'istruzione LDIR, si copia una locazione di memoria, il cui valore è 00_H, nella successiva per 2047 volte (BC volte) spostandosi ogni volta di una locazione.

Quindi viene definita l'area di stack all'indirizzo F00, cioè all'inizio dell'ultima pagina RAM. Il registro indice IX viene caricato, per comodità successive, con l'indirizzo di partenza della RAM (800). Viene poi predisposto il modo di interruzione 2. Tale modo, il più complesso e flessibile dei 3 possibili sul μP Z80, richiede che all'atto della generazione del segnale INT il periferico che ha richiesto l'interruzione ponga sul bus dati del sistema un byte che viene chiamato «vettore di interruzione». Tale byte, acquisito dalla CPU all'atto del riconoscimento dell'interruzione , costituisce la parte bassa di un indirizzo a 16 bit la cui parte alta è costituita dal registro I della CPU. Questo indirizzo va a puntare una locazione di una tabella in memoria alla quale si trova l'indirizzo della routine di gestione dell'interruzione. Quindi si carica in I (registro interruzione) il valore 09 che costituirà come detto la parte alta dell'indirizzo a cui prelevare l'indirizzo della routine per la gestione dell'interruzione; da notare che questa tabella è in RAM, ma poteva anche essere in ROM. Successivamente si carica IY con l'indirizzo della routine di interruzione che è 0500 (la routine è quindi in ROM) e poi lo si ricopia all'indirizzo 0920. Facendo ciò si è stabilito che se il periferico fornisce come vettore di interruzione il byte 20, la CPU si costruisce l'indirizzo I→09 + →20→ →0920 dal quale preleva l'indirizzo 0500 ed esegue un salto a quest'ultimo dove troverà la routine di interruzione. Le due istruzioni successive difatti eseguono l'out del byte 20 sulla porta A-controllo del PIO, codice dispositivo 46; il bit b_0 a livello logico basso informa la porta A del PI0 che tale byte è il suo nuovo vettore di interruzione. Similmente si predispone quindi la porta A al funzionamento in modo 1, o modo di ingresso, con l'invio del byte 01001111 in cui i bit $b_0 \div b_3$ ad 1 informano la porta che i due bit più significativi del byte selezionano il modo di ingresso. In maniera analoga si programma la porta B per il modo di uscita con l'invio al periferico 47 (Porta B-Controllo) del byte 00001111. Infine l'invio sulla porta A-controllo del byte 10000111 viene riconosciuto dal PIO come parola di controllo delle interruzioni ($b_0 \div b_3 \rightarrow 0111$) ed il bit b_7 ad 1 abilita la porta A a generare richieste di interruzione sulla linea $\overline{INT}$. Da notare che per la porta B del PIO non è stato programmato il vettore di interruzione nè sono state abilitate le interruzioni in quanto essendo programmate in modo 0 (uscita) non ha necessità di chiedere l'attenzione della CPU per comunicarle alcunché.

2. Programma principale

Terminate le inizializzazioni «di sistema» si passa alle inizializzazioni proprie del programma. Il registro H, che come si vedrà in seguito è un contatore che indica quante volte si può ancora facilitare il giocatore aumentando la durata del delay, viene posto a 5. Le locazioni di memoria IX + 3, + 4, e + 5 (cioè 803, 804 e 805) vengono caricate con 02, 00 e FF rispettivamente; tali locazioni determinano la durata del ritardo della subroutine di delay. Il registro B viene posto a 2; tale registro indica il numero di errori che si possono fare prima che il programma decida di aumentare la durata del ritardo. Successivamente viene caricato il registro C con il valore FE (1111 1110); questo registro contiene in pratica il valore che deve essere replicato dal giocatore.

A questo punto si entra nel ciclo principale. Il registro C viene copiato in A per il successivo OUT. Vengono abilitate le interruzioni mascherabili e successivamente viene inviato il valore di C, ora in A, alla porta B del PIO (modo OUT). Viene chiamata la routine di delay che genera un ritardo di una certa durata. Al ritorno dalla routine vengono disabilitate immediatamente le interruzioni mascherabili, e si procede a cambiare il contenuto del registro C, in questo caso con una semplice rotazione circolare a sinistra. Quindi si rientra nel ciclo principale. Da notare la posizione delle istruzioni EI e DI: questa è tale che l'interruzione è presa in considerazione solamente durante la chiamata e lo svolgimento ed il ritorno dalla routine delay.

Ciò non significa che le richieste di interruzione generate al di fuori di questo lasso di tempo cadano nel nulla. Difatti occorre ricordare che si ha a che fare con un dispositivo PIO, il quale è dotato di un latch «richiesta di interruzione pendente» che continua a richiedere la attenzione della CPU fintanto che essa non la soddisfi o il PIO venga resettato. Ciò, anche se è improbabile che si verifichi, data l'esiguità del tempo impiegato, torna a discapito del giocatore poiché, nel caso, l'accettazione dell'interruzione avviene durante il nuovo ciclo EI→DI, prima del quale però il registro C ha subito un cambiamento.

3. Routine di gestione dell'interruzione

Ogni qualvolta il giocatore genera un segnale $\overline{\text{ASTB}}$, la porta A del PIO acquisisce lo stato degli switches e genera una richiesta di interruzione alla CPU in attesa che venga soddisfatto. Quando ciò accade la prima cosa che la CPU fà in risposta alla richiesta del PIO, oltre a disabilitare le interruzioni, è di leggere il contenuto appena immesso nel registro della porta A, cioè lo stato degli switches, quindi lo confronta col valore di C per vedere se il giocatore ha immesso il valore esatto. È importante osservare che il valore dato dal giocatore deve essere l'esatto complemento di quello che egli vede sui led. Questo perché l'istruzione OUT del ciclo principale non è preceduta da CPL quindi è il valore di C che pilota i led, i quali si accen-

dono così in presenza dello 0 logico del registro C.

Nel caso che il giocatore abbia indovinato il programma entra in loop dal quale si esce solamente o col RESET del sistema o con una richiesta $\overline{NMI}$, e visualizza alternativamente il valore di C ed il suo complemento.

Diversamente decrementa il registro B e, se questi non è zero (DJNZ), rientra dalla interruzione. Se B è 0 vuol dire che si è esaurito il numero di errori permessi per quella data durata di delay, (o velocità di gioco che dir si voglia) quindi si decrementa il contatore H e se questi non è zero si aumenta la durata del delay aumentando il valore della locazione 804 e si ritorna al programma principale. L'uso di un'istruzione RETI, invece di una normale RET, serve per informare il dispositivo PIO che è appena terminata la gestione di una richiesta di interruzione in modo che il PIO aggiorni la sua logica di controllo in maniera opportuna. Se invece H, dopo che è stato decrementato, è zero, allora significa che il giocatore ha esaurito tutte le sue 10 ($B \times H = 5 \times 2 = 10$) possibilità e non gli è più concesso un altro aumento del ritardo in input. Il programma esegue una restart alla locazione 10_H ove si trova l'istruzione HALT che ferma l'elaborazione in attesa di una richiesta di interruzione. Dato però che le interruzioni sono rimaste disabilitate dalla chiamata della routine di interruzione, l'unico modo per uscire da questo vicolo cieco è quello di generare manualmente o un «sistem reset» oppure una $\overline{NMI}$ la quale porterà il controllo del processo alla locazione 66. Da questa, presupponendo la EPROM riempita coi valori 00 nei suoi spazi non utilizzati, con una serie di NOP la CPU giungerà al ramo di inizializzazione alla locazione 70.

Da notare che la mancanza dell'istruzione EI prima della RETI preclude la possibilità di generare e far servire due interruzioni durante la stessa chiamata della subroutine delay. Le interruzioni vengono infatti riabilitate solo al prossimo ciclo del programma principale.

4. Routine di delay

Questa semplice routine di delay è composta da tre loop nidificati. Il caricamento dei tre registri contatori è effettuato mediante l'uso del registro IX in base alle locazioni di memoria 803, 804 e 805. Tuttavia solo la locazione 804 viene alterata per modificare la durata del ritardo. Difatti la durata di tale ritardo è di circa 0,5 secondi in condizioni iniziali, successivamente dopo ogni aumento, la sua durata sarà di 0,098, 0,19, 0,22 e 0,4 secondi. I tempi ovviamente sono calcolati per una frequenza di clock di 4 MHz.

Occorre notare che poiché il valore iniziale del contatore del 2° ciclo è 0; data la posizione dell'istruzione DEC D il ciclo non viene eseguito 0 volte bensí 255 + 1 volte perché al primo blocco decisionale il contatore arriva non con il valore 0 ma $0\text{-}1 = FF_H$.

Il programma in assembly, realizzato mediante il sistema di sviluppo UX8/22 della SGS e caricato in EPROM, è qui di seguito riportato.

```
ln  addr  obj code t e i line   source statement

 1                          1   ;              INIZIALIZZAZIONE SISTEMA
 2                          2
 3                          3   RAM:    .EQU 0800H        ;RAM=0800H
 4                          4           .ORG 0000H        ;IND. ALLOCAZIONE
 5  0000  AF                5           XOR  A            ;AZZERA A
 6  0001  320008            6           LD   (RAM),A      ;AZZERA LOC. 0800H
 7  0004  210008            7           LD   HL,RAM       ;INIZIALIZZA HL
 8  0007  110108            8           LD   DE,RAM+1     ;INIZIALIZZA DE
 9  000A  01FF07            9           LD   BC,07FFH     ;N.LOC.DA AZZERARE
10  000D  EDB0             10           LDIR              ;AZZERA TUTTA LA RAM
11  000F  31000F           11           LD   SP,0F00H     ;DEFIN. AREA STACK
12  0012  DD210008         12           LD   IX,RAM       ;PUNTA 1' LOC. RAM
13  0016  ED5E             13           IM   2            ;SEL. MODO INT. 2
14  0018  3E09             14           LD   A,09H        ;IND. HI TAB. INT.
15  001A  ED47             15           LD   I,A
16  001C  FD210005         16           LD   IY,0500H     ;IND. ROUTINE INT.
17  0020  FD222009         17           LD   (0920H),IY   ;CARICAM. TAB. INT.
18  0024  3E20             18           LD   A,20H        ;VETT. INT. PIO A
19  0026  D346             19           OUT  (46H),A
20  0028  3E4F             20           LD   A,4FH        ;PIO A MODO 1 (IN)
21  002A  D346             21           OUT  (46H),A
22  002C  3E0F             22           LD   A,0FH        ;PIO B MODO 0 (OUT)
23  002E  D347             23           OUT  (47H),A
24  0030  3E87             24           LD   A,87H        ;ABILITA INT. PIO A
25                         25
26                         26
27                         27   ;              PROGRAMMA PRINCIPALE
28                         28
29  0032  2605             29           LD   H,05H        ;NUM. FACILITAZIONI
30  0034  DD360302         30           LD   (IX+3),02H   ;INIZIALIZZ. DELAY
31  0038  DD360401         31           LD   (IX+4),01H
32  003C  DD3605FF         32           LD   (IX+5),0FFH
33  0040  0602             33           LD   B,02H        ;NUMERO CHANCES
34  0042  0EFE             34           LD   C,0FEH       ;CONFIGURAZ. PER LED
35  0044  79               35   ROUT:   LD   A,C
36  0045  FB               36           EI                ;ABILITA INT. CPU
37  0046  D345             37           OUT  (45H),A      ;VISUALIZZA SUI LED
38  0048  CD5000           38           CALL DELAY        ;GENERA IL RITARDO
39  004B  F3               39           DI                ;DISABILITA INT. CPU
40  004C  CB01             40           RLC  C            ;NUOVA CONFIGURAZ.
41  004E  18F4             41           JR   ROUT         ;RIPETE
42                         42
43                         43
44                         44   ;              SUBROUTINE DELAY
45                         45
46  0050  DD6E03           46   DELAY:  LD   L,(IX+3)     ;PARAMETRI DI DELAY
47  0053  DD5604           47   DL3:    LD   D,(IX+4)
48  0056  DD5E05           48   DL2:    LD   E,(IX+5)
49  0059  1D               49   DL1:    DEC  E
50  005A  20FD             50           JR   NZ,DL1       ;LOOP INTERNO
51  005C  15               51           DEC  D
52  005D  20F7             52           JR   NZ,DL2       ;LOOP CENTRALE
53  005F  2D               53           DEC  L
54  0060  20F1             54           JR   NZ,DL3       ;LOOP ESTERNO
55  0062  C9               55           RET               ;RIENTRO
56                         56
57                         57
```



```
ln  addr  obj code t e i line   source statement

 1                          58   ;       SUBROUTINE GESTIONE INTERRUZIONE
 2                          59
 3                          60           .ORG 0500H        ;IND. ROUTINE INT.
 4  0500  DB44             61           IN   A,(44H)      ;LETTURA SWITCHES
 5  0502  B9               62           CP   C            ;COMPARA L'ORIG.
 6  0503  2812             63           JR   Z,OKAY       ;ALLA VITTORIA
 7  0505  100E             64           DJNZ SBA          ;CICLO CHANCES
 8  0507  25               65           DEC  H            ;DEC. FACILITAZ.
 9  0508  2001             66           JR   NZ,LENTO     ;ALLA FACILITAZ.
10  050A  76               67           HALT              ;ARRESTO TOTALE
11  050B  DD7E04           68   LENTO:  LD   A,(IX+4)     ;VAL.CENTRALE DELAY
12  050E  C630             69           ADD  A,30H        ;AUMENTO DELAY
13  0510  DD7704           70           LD   (IX+4),A     ;AGGIORNA IN RAM
14  0513  0602             71           LD   B,02H        ;ALTRE 2 CHANCES
15  0515  ED4D             72   SBA:    RETI              ;RIENTRO DALL'INT.
16                         73
17  0517  2F               74   OKAY:   CPL               ;COMPLEMENTA A
18  0518  D345             75           OUT  (45H),A      ;VISUALIZZA SUI LED
19  051A  CD5000           76           CALL DELAY        ;GENERA IL RITARDO
20  051D  18F8             77           JR   OKAY         ;RIPETE
21                         78
22                         79
23                         80
24                         81           .END
    0 error(s)
```

SOLUZIONI ESERCIZI PROPOSTI

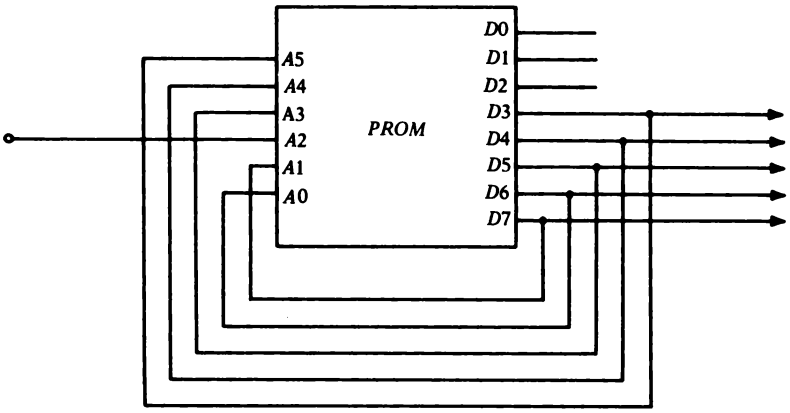
CAPITOLO PRIMO

1. per f_4 all'indirizzo 172;
   per f_3 agli indirizzi 17, 19, 21, 23, 49, 51, 53, 55, 81, 83, 85, 87, 113, 115, 117, 119.
   per f_2 agli indirizzi 160, 161, 162, 163, 164, 165, 166, 167, 224, 225, 226, 227, 228, 229, 230, 231.
   per f_1 agli indirizzi 0, 1, 2, 3, 11, 28, 30, 32, 33, 34, 35, 37, 39, 44, 46, 64, 65, 66, 67, 76, 77, 78, 79, 92, 93, 94, 95, 96, 97, 98, 99, 101, 103, 108, 110, 156, 158, 160, 161, 162, 163, 165, 167, 172, 174, 194, 196, 198, 202, 204, 205, 206, 207, 210, 212, 213, 214, 218, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 234, 236, 237, 238, 242, 244, 245, 246.

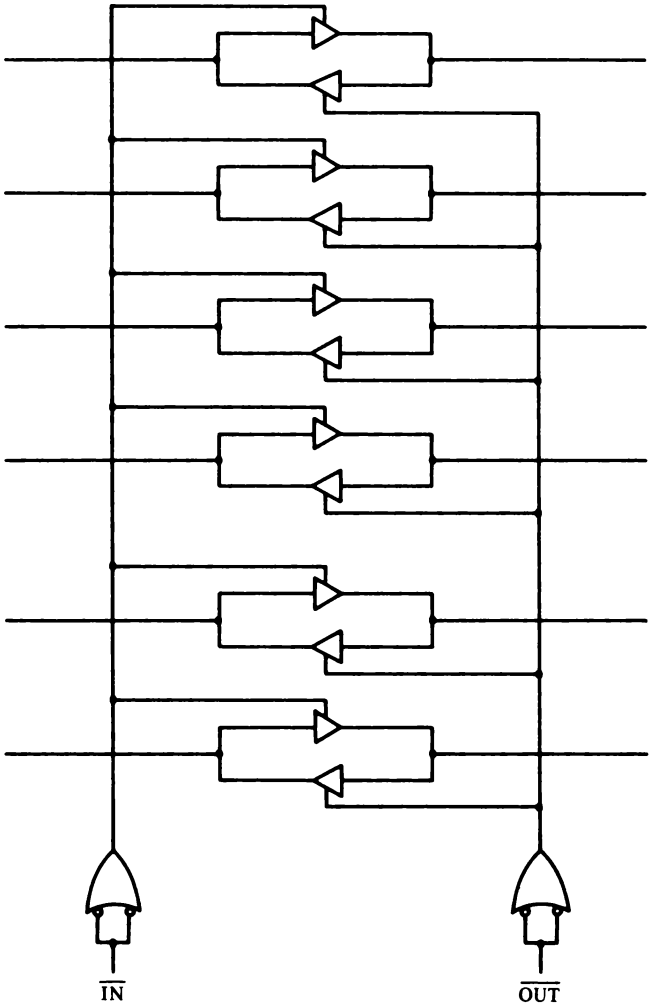
2.

INGRESSO						USCITA			
A	B	C	D	E	F	f_1	f_2	f_3	f_4
1	0	0	1	X	X	1	0	0	0
1	X	0	X	X	X	1	0	0	0
X	1	1	1	X	X	1	0	0	1
0	X	0	X	X	1	1	0	0	0
X	X	X	0	1	1	0	1	0	0
1	0	X	1	X	X	0	1	0	0
1	X	X	X	1	1	0	1	0	0
0	0	1	1	X	X	0	0	1	0
1	0	1	1	1	0	0	0	1	0
1	X	0	0	X	X	0	0	0	1
X	0	X	X	0	0	0	0	0	1

3.



4.



5.

STATI		Y_1	Y_2	Y_3	Y_4
1	2	1	1	1	1
3	**2**	1	1	1	0
3	4	1	1	0	0
5	**4**	1	0	0	0
5	6	0	0	0	0
7	**6**	0	0	0	1
7	8	0	0	1	1
9	**8**	0	0	1	0
9	10	0	1	1	0
1	**10**	1	1	1	0

uscite con frequenza
di ck divisa per cinque

CAPITOLO SECONDO

1. NO, perchè l'istruzione POP DE deve precedere POP HL (l'ultimo ad entrare deve essere il primo ad uscire: «LIFO»).

2. Perché lo stack *mangia* una parte del programma.

3.

0102	21 BB AA	LD HL, AABB
0105	22 00 01	LD (0100), HL
0108	76	HALT

4.

	0181	3E 60	LD A, 60
P_3	0183	06 60	LD B. 60
P_2	0185	0E 60	LD C, 60
P_1	0187	OD	DEC C
	0188	C2 87 01	JPNZ P_1
	018B	05	DEC B
	018C	C2 85 01	JPNZ P_2
	018F	3D	DEC A
	0190	C2 83 01	JPNZ P_3

5.

0193	3E n_1	LD A, n_1
0195	06 n_2	LD B, n_2
0197	A0	AND B
0198	57	LD D,A
0199	76	HALT

6.

019A....	01A3	byte memorizzati	
Programma:			
01A4	0E 20	LD C,20	
01A6	06 0A	LD B, 0A	
01A8	21 9A 01	LD HL, 019A	
01AB	ED B3	OTIR	
01AC	76	HALT	

7.

01AD....	01B6	dati da trasferire	
Programma:			
01B7	21 AD 01	LD HL, nn	
01BA	11 00 04	LD DE, nn	
01BD	01 0A 00	LD BC, nn	
01C0	ED B0	LDIR	carica (HL) in (DE), incrementa DE ed HL. Decrementa BC e ripeti fino a quando BC = 0
01C2	76	HALT	

8.

01C3....	01D2		zona di memoria in cui sono caricati i 16 numeri esadecimali
Programma:			
P_1 01D3	21 C3 01	LD HL, nn	
01D6	11 C4 01	LD DE, nn	
01D9	06 0F	LD B	
P_2 01DA	1A	LD A, (DE)	
01DB	BE	CP (HL)	confronta A con (HL)
01DC	DC E7 01	CALL SCAMBIO	se C = 1
01E0	38 F1	JRC P_1	salta a PC+ e se il flag di carry = 1
01E2	23	INC HL	
01E3	13	INC DE	
01E4	10 F5	DJNZ P_2	decrementa B e salta a PC+e se B = 0
01E6	FF	RST 38 H	
scambio:			
01E7	4E	LD C, (HL)	
01E8	77	LD (HL), A	
01E9	79	LD A, C	
01EA	12	LD (DE), A	
01EB	C9	RET	

9.

01E7 (moltiplicando)	01EB (risultato)
01E8	01EC
01E9 (moltiplicatore)	
01ED	

Programma:

	Locazione di memoria	Codice oggetto	Assembly	Commenti
	01ED	21 00 00	LD HL, 0000	azzera HL
	01F0	ED 5B E7 01	LD DE, (01E7)	carica il moltiplicando in DE
	01F4	ED 4B E9 01	LD BC, (01E9)	carica il moltiplicatore in BC
	01F8	7A	LD A, D	
	01F9	B3	OR E	
	01FA	20 01	JRNZ, P_1	salta a PC+e se Z = 0
	01FC	FF	RST 38 H	
P_1	01FD	CB 38	SRL B	shifta a destra B
	01FF	CB 19	RR C	ruota a destra attraverso il carry il contenuto di C
	0201	30 04	JRNC, P_2	salta a PC+e se il flag di carry è uguale a zero
	0203	19	ADD HL, DE	somma HL con DE e poni il risultato in HL
	0204	30 01	JRNC, P_2	
	0206	FF	RST 38H	
P_2	0207	78	LD A,B	
	0208	B1	OR C	
	0209	CA 16 02	JP Z, P_4	memorizza il risultato se il contenuto di BC = 0
	020C	CB 23	SLA E	shifta aritmeticamente E verso destra
	020E	CB 12	RL D	ruota a sinistra attraverso il carry il registro D
	0210	30 01	JRNC, P_3	esiste overflow?
	0212	FF	RST 38 H	se esiste overflow il programma si arresta in questo punto, altrimenti continua fino a 0219
P_3	0213	C3 FD 01	JP P_1	
P_4	0216	22 EB 01	LD (01EB), HL	poni il risultato nella locazione di memoria scelta
	0219	FF	RST 38H	

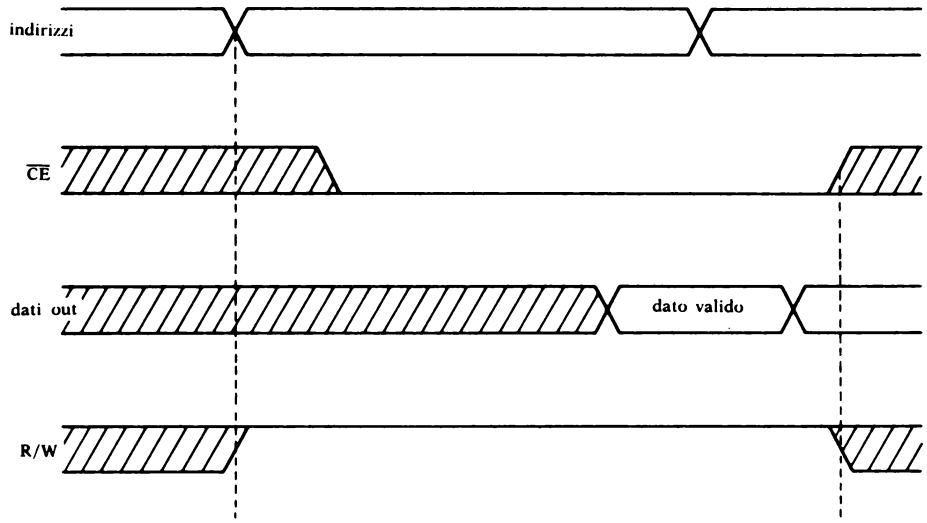
10.

	0220	3E CO	LD A, C0	
	0222	06 00	LD B, 00	
	0224	0E 0C	LD C, 0C	
	0226	CB 3F	SRL A	
P_1	0228	04	INC B	
	0229	B9	CP C	
	022A	C2 28 02	JPNZ P_1	
	022D	76	HALT	

CAPITOLO TERZO

1. Per $Q_A = 1\ \mu s$; per STB = 927 ns.

2.



3. Acceso

4.

codice istruzione	microistruzioni	segnali di controllo E_{N1} E_{N2} E_{N3}
IR = 00	ALT	0 0 0
IR = 10	$\overline{B} \rightarrow B$	0 1 0
	$B + 1 \rightarrow B$	0 0 1
	$A + B \rightarrow A$	1 0 0
IR = 11	se F = 0 esegui	
	$A + B \rightarrow A$	0 0 1
	se F = 1	
	ALT	0 0 0

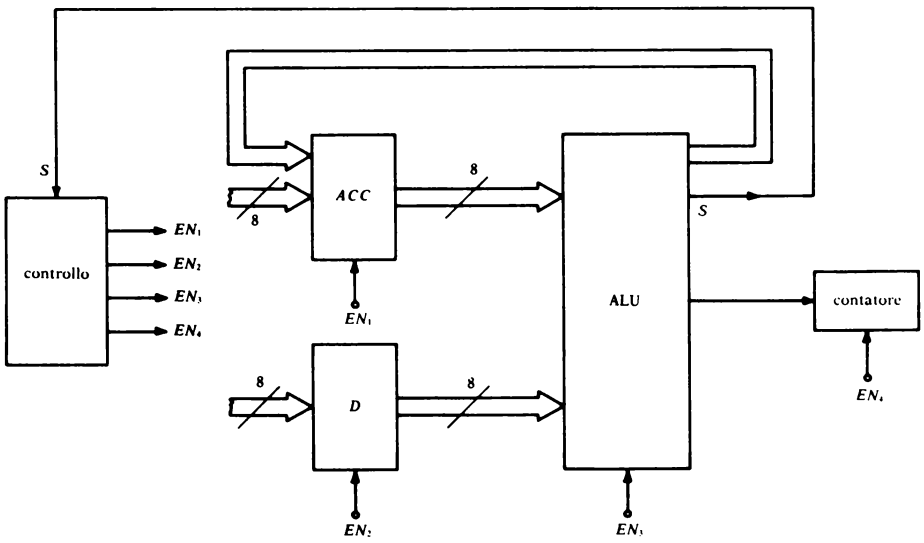
5.

stato attuale	prossimo stato	comandi					
		E_{N1}	E_{N2}	E_{N3}	E_{N4}	E_{N5}	E_{N6}
0	Y	1	0	0	0	0	0
Y	A	0	1	0	1	0	0
A	B	1	0	1	0	1	0
B	C	0	1	0	1	0	1
C	B	1	0	1	0	1	1

La variabile aggiuntiva E_{N6} è necessaria per distinguere i due stati A e C che altrimenti sarebbero uguali.

6.

Supponendo che nel registro ACC si trovi il dividendo e che nel registro D il divisore, uno fra i possibili schemi a blocchi può essere quello della figura seguente nella quale:



EN_1 trasferisce il contenuto di ACC in ALU;
EN_2 trasferisce il contenuto di D in ALU;
EN_3 effettua la sottrazione ACC - D se ACC > D e mette il risultato in ACC.
EN_4 incrementa il contenuto del registro CONT che conterrà il risultato della divisione.

Il segnale S emesso dalla ALU, avverte il controllo di terminare le operazioni quando ACC è diventato minore di D.

7.

Le fasi di fetch, sempre uguali per le cinque istruzioni del programma, sono le seguenti:

1) il contenuto del PC, vale a dire l'indirizzo della istruzione che deve essere eseguita, viene posto sul bus degli indirizzi;

2) il contenuto della locazione di memoria indirizzato dal PC viene posto sul registro IR;

3) il PC viene incrementato di uno.

Le prime tre fasi di execute sono simili poiché relative alla stessa operazione LD dd, nn e sono composte dai seguenti cicli:

1) il contenuto del secondo byte dell'istruzione viene posto nella parte bassa dei due registri;

2) il PC viene incrementato di uno;

3) il contenuto del terzo byte dell'istruzione viene posto nella parte alta dei due registri;

4) il PC viene incrementato di uno.

La fase di execute per l'istruzione LDIR è la seguente:

1) (DE) ← (HL)

2) DE ← DE + 1

3) HL ← HL + 1

4) BC ← BC — 1

5) se BC = 0 l'istruzione è terminata ed il PC si incrementa di due, altrimenti continua con i primi quattro cicli

La fase di execute dell'istruzione HALT consiste nella esecuzione di una serie di NOP che hanno lo scopo di mantenere il refresh di eventuali memorie.

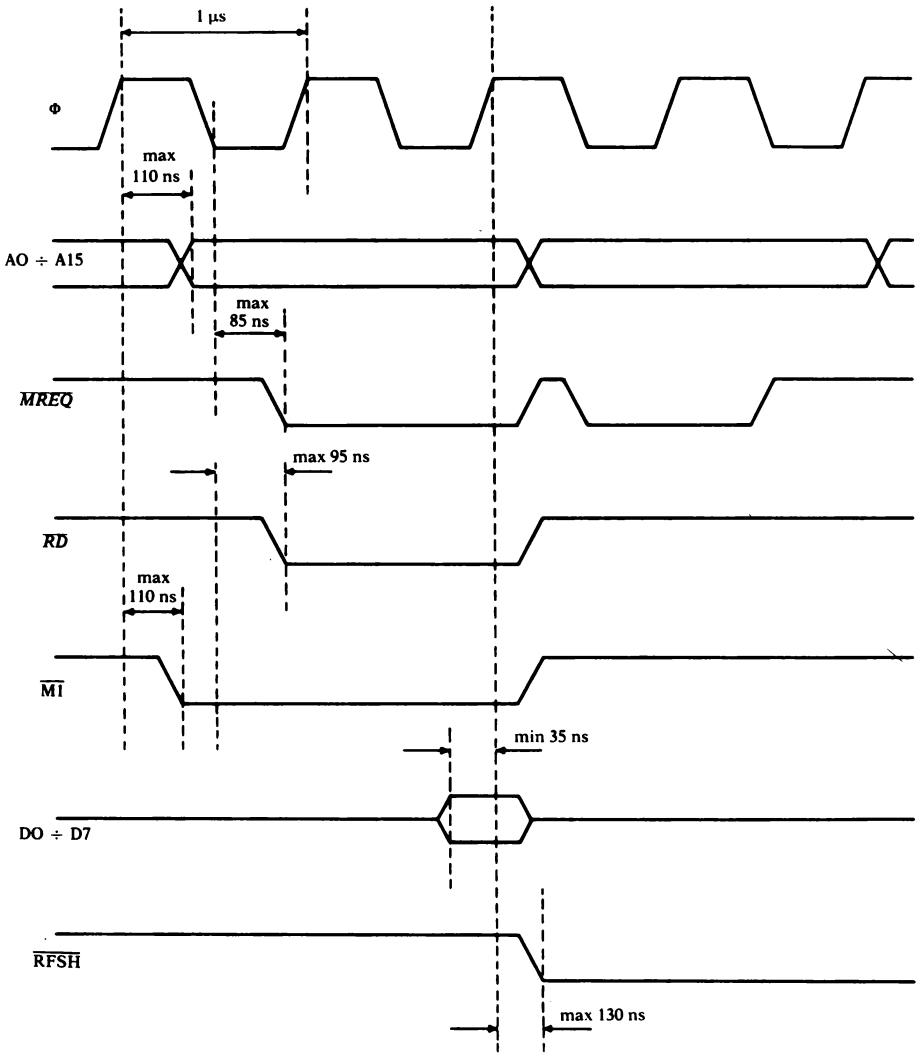
8.

I tempi indicati sono misurati in ns. Il diagramma è a pagina seguente.

9. Il ciclo di lettura è comandato dalle seguenti temporizzazioni:

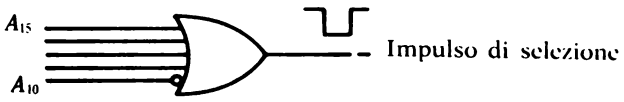
$t_{D(AD)}$ = max 110 ns
$t_{DL}\Phi_{(MR)}$ = max 85 ns;
$t_{DL}\Phi_{(RD)}$ = max 95 ns;
$t_{S}\Phi_{(D)}$ = min 50 ns;
$t_{DH}\Phi_{(MR)}$ = max 85 ns;
$t_{DH}\Phi_{(RD)}$ = max 85 ns.

10. Almeno 48 ns.

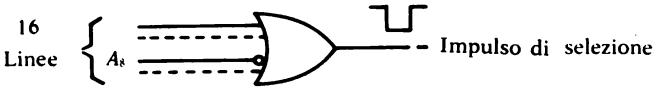

1 μs
Φ
max
110 ns
AO ÷ A15
max
85 ns
MREQ
max 95 ns
RD
max
110 ns
M1
min 35 ns
DO ÷ D7
RFSH
max 130 ns

CAPITOLO QUARTO

1.



2.



3. Non è possibile

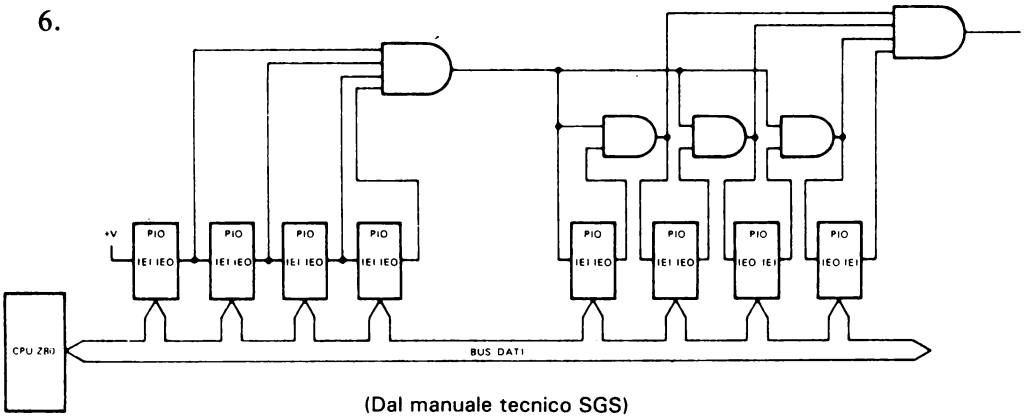
4. 128 nel caso di I/0 isolato. Dipende dalla memoria fittizia assegnata nel caso di I/0 Memory Mapped.

5.

0	1	0	1	0	1	1	1

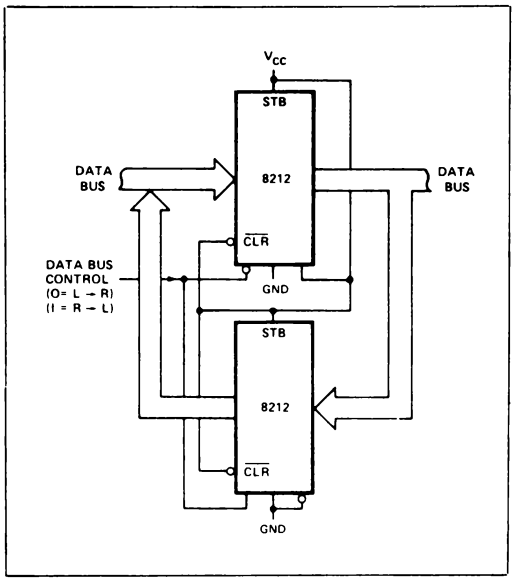
Si avrà una richiesta di interruzione non appena i sensori collegati alle linee di A_7, A_5 ed A_3 porteranno simultaneamente tali linee dal livello logico basso al livello logico alto.

6.



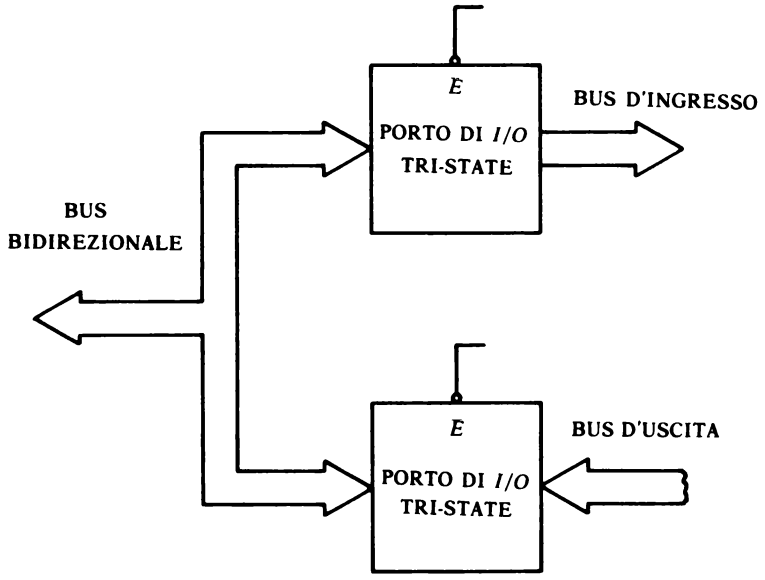
(Dal manuale tecnico SGS)

7.

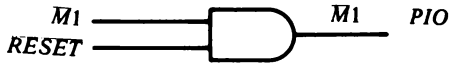


(Dal manuale tecnico INTEL)

8.



9.



10.

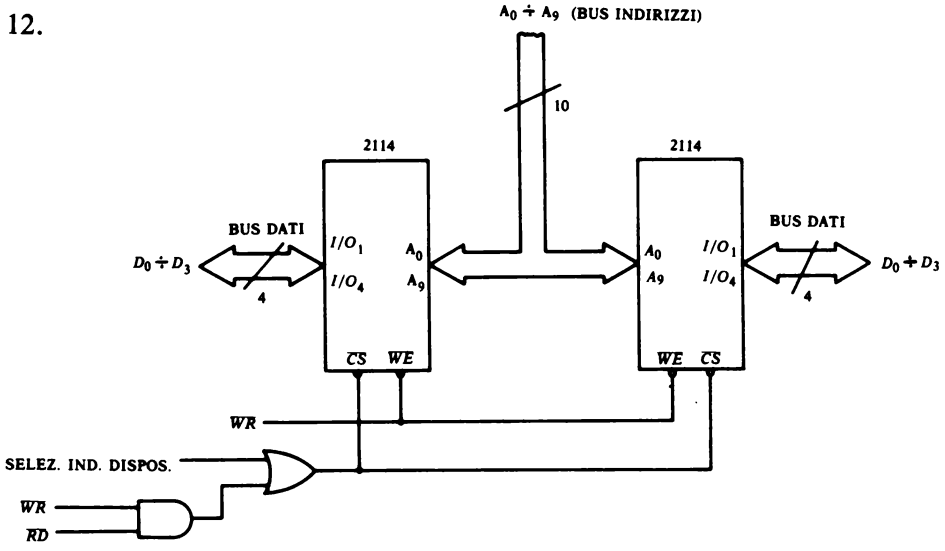
TAB IN	TAB OUT
00	FF
FF	00

11.

	0100	31 00 0A
		11 00 03
		21 00 04
		06 04
P_1	010B	1A
		32 00 40
		3A 01 40
		E6 0F
		BE
		C2 22 01
		13
		23
		10 EF
		3E 00
P_2	011E	32 00 08
		FF
		3E 01
		C3 1E 01

	TAB IN	TAB OUT
0300		0400

12.

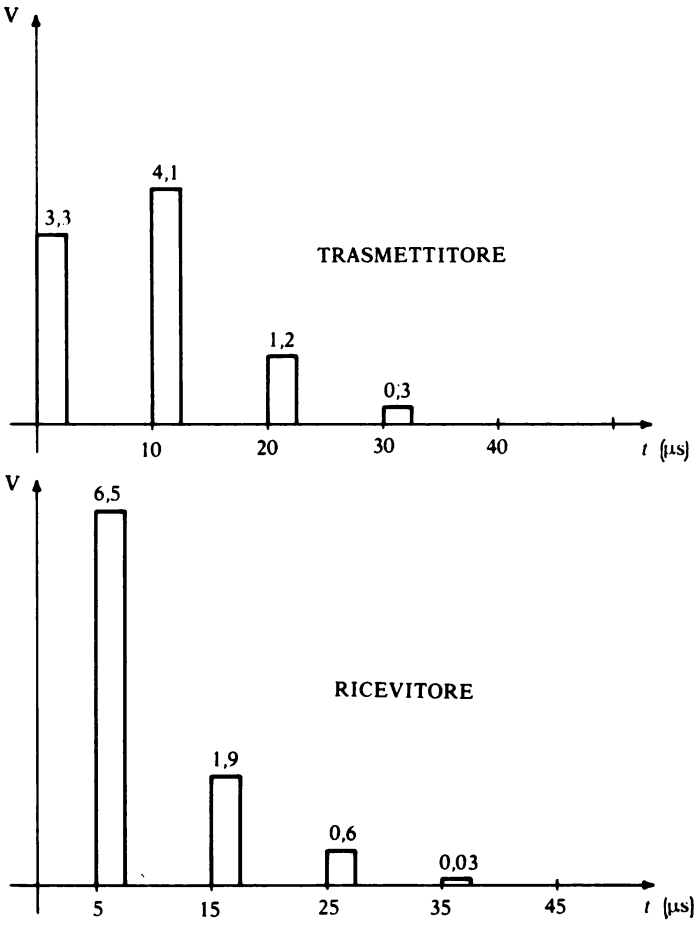


CAPITOLO QUINTO

1. No, perché i *drivers* non abilitati cimano le eventuali sovratensioni.

2. $$Z_0' = \frac{120}{\sqrt{1 + \dfrac{5 \times 15 \times 10^{-12} + 10 \times 10^{-12}}{30 \times 10^{-12} \cdot 0{,}5}}}$$

3. $$K_L = \frac{52{,}5 - j\,19{,}8 - 120}{52{,}5 - j\,19{,}8 + 120}$$

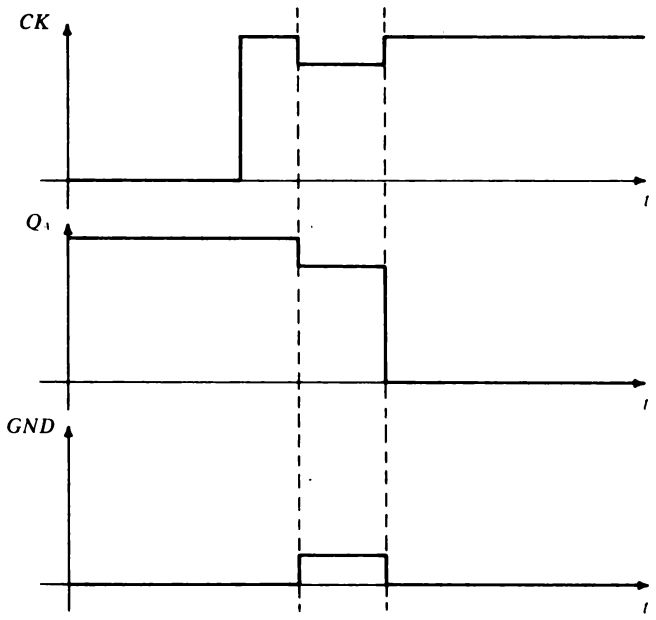
4. 

5. Per l'estremità trasmittente i primi tre livelli valgono rispettivamente : 0,66 V, 1,03V, 0,93V.
Per l'estremità ricevente: 1,21V, 0,88V, 0,97V.

6. Per l'estremità trasmittente i primi tre livelli valgono rispettivamente: 0,166V, 0,35V, 0,46V.
Per l'estremità ricevente: 0,28V, 0,44V.

7. Supponendo che le porte TTL siano del tipo 7400, le tensioni successive all'uscita del driver sono: 1,3V, 2,4V, 3,6V. Per il ricevitore sono: 2,75V, 3,6 V.

8.



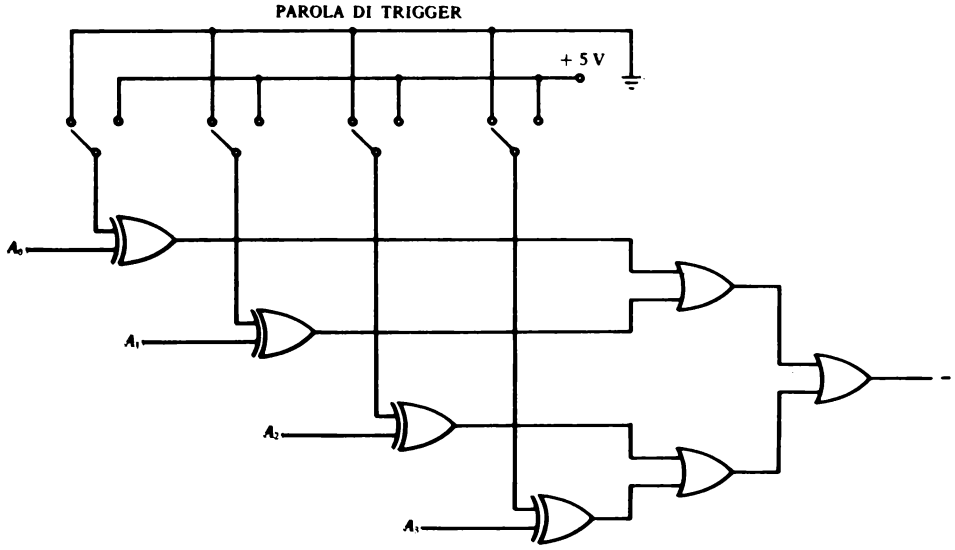
9. Circa 20 cm.

10. Circa 40 m.

11. 1 Km.

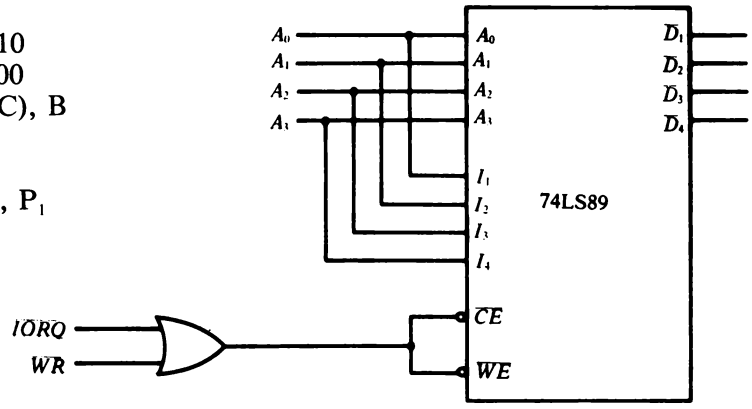
CAPITOLO SESTO

1.



2.

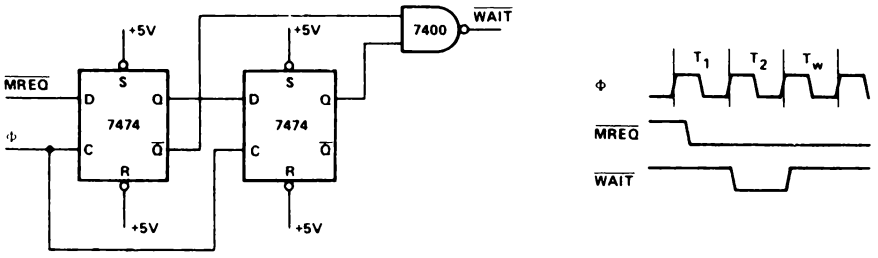
```
        LD A,10
        LD C,00
P1      OUT (C), B
        INC C
        CP C
        JP NZ, P1
        HALT
```



3.

```
                OUT (40), A          motore avanti
                CALL RITARDO
                OUT (n), A           impulso di passo
                CALL RITARDO
                OUT (50), A          motore indietro
                CALL RITARDO
                OUT (n), A           impulso di passo

RITARDO         LD B, FF
P1              DEC B
                JP NZ, P1
                RET                  con n = 10,20,30
```

4.



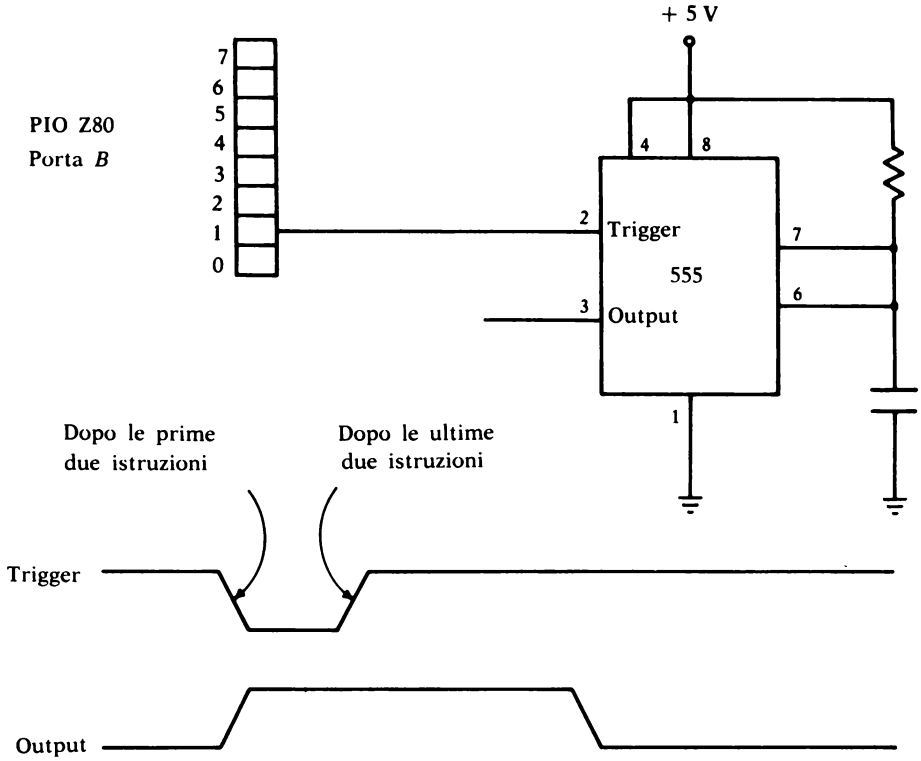
Questo circuito permette l'utilizzo da parte del microcomputer di memorie lente e quindi a più basso costo (Dal manuale tecnico SGS).

5.

Software:

```
LD A,00          Carica 0 nell'accumulatore
OUT (Port B), A  Mettilo in uscita sulla porta
                 B del PI0
LD A,02          Carica 1 nel bit 2 dell'accu-
                 mulatore
OUT (Port B),A   Mettilo in uscita sulla porta
                 B del PI0
```

Hardware:



6. Sì

GLOSSARIO

Termini usati nei sistemi a microprocessore

Access Time (Tempo di accesso)
Intervallo di tempo tra la richiesta dell'informazione e l'istante in cui l'informazione è disponibile o valida.

Accumulator (Accumulatore)
Registro di uso generale usato per operazioni aritmetiche e logiche.

ACK-Acknowledgement (Riconoscimento)
Risposta di un elemento per indicare che la richiesta è stata accettata.

ADC
Convertitore analogico digitale.

Adder (Sommatore)
Circuito logico che combina i bit per generare la somma ed il riporto di questi bit.

Address (Indirizzo)
Espressione, di solito numerica, che individua una data locazione di memoria.

Address bus (Bus indirizzi)
Insieme di linee che vengono inviate a più moduli di memoria o a dispositivi di I/0, per indirizzarli.

Address Register (Registro Indirizzo)
Registro in cui è immagazzinato l'indirizzo.

Algorithm (Algoritmo)
Termine matematico usato per descrivere un insieme di regole ben definite o procedimenti per la soluzione di un problema o di un programma.

Alphanumeric Code (Codice Alfanumerico)
Codice consistente in un insieme di lettere, cifre, e caratteri speciali.

ALU (Unità Aritmetico Logica)
Sottosistema di calcolo, in un sistema digitale, in grado di eseguire operazioni matematiche.

Analog Representation (Rappresentazione Analogica)
Rappresentazione di una grandezza continuamente variabile che non possiede valori discreti.

Application Software (Software Applicativo)
Programmi scritti nel linguaggio del computer per realizzare una determinata applicazione.

Architecture (Conformazione)
Struttura hardware di base di un computer.

Arithmetic Shift (Shift Aritmetico)
Uno scorrimento che non influisce sul bit di segno.

Array Logic (Logica a schiera)
Rete logica la cui configurazione consiste in un rettangolo ordinato di intersezione di fili d'ingresso-uscita, con elementi connessi ad una di queste intersezioni. Di solito è un codificatore o un decodificatore.

ASCII
Codice standard nazionale americano utilizzato per l'interscambio delle informazioni. Consiste in un set di caratteri grafici e di controllo.

Assembler
Programma che traduce programmi scritti in linguaggio simbolico in codici del linguaggio macchina.

Assembly
Linguaggio in cui le istruzioni in codice macchina vengono sostituite con codici mnemonici e gli indirizzi con delle labels.

Asynchronous Device (Dispositivo Asincrono)
Dispositivo in cui la velocità di operazione non è correlata alla frequenza del sistema ad esso connesso.

Autoload (Caricamento automatico)
Possibilità hardware che permette il caricamento di piccoli programmi di bootstrap.

Background Processing
Esecuzione automatica di un programma a bassa prorità quando programmi a più alta priorità non stanno utilizzando le risorse del sistema.

Bandwidth (Larghezza di banda)
Differenza tra le due frequenze di taglio di una banda.

Basic
Linguaggio ad alto livello abbreviazione di «Beginners All-Purpose Simbolic Instruction Code».

Batch
È detto così un sistema di elaborazione dati in cui un certo numero di programmi sono eseguiti sequenzialmente dal computer.

Baud
Velocità di trasmissione di simboli al secondo, in cui il simbolo è definito come numero di bit tramessi simultaneamente. Di solito coincide con la velocità di trasmissione espressa come bit al secondo.

Baudot Code (Codice Baudot)
Codice di informazione usato nella trasmissione dati.

Baude Rate
Velocità alla quale il dato è trasmesso attraverso una linea di collegamento. Per linee asincrone il baud-rate è uguale a circa dieci volte il numero di caratteri per secondo.

Benckmark
Programma di test per misurare la velocità o l'efficienza di un computer.

Binary Code (Codice Binario)
Codice che usa due distinti caratteri, di solito i numeri 0 ed 1.

Binary Coded Decimal (Codice Binario Decimale o BCD)
Sistema di numerazione binario, per la codifica di numeri decimali, utilizzante quatto bit.

Bit
Cifra binaria.

Block (Blocco)
Gruppo di dati o di parole trattati come una singola unità.

Block Diagram (Diagramma a blocchi)
Diagramma di un sistema, strumento o computer, in cui le parti principali sono rappresentate, da convenienti figure geometriche, utilizzato per mostrare sia le funzioni principali sia le relazioni funzionali fra le singole parti.

Bootstrap
Piccolo programma usato per caricarne un altro più grande e di solito più complesso.

Borrow (Prestito)
Riporto aritmetico negativo.

Branching
Metodo per selezionare, basandosi sui risultati, la prossima operazione da eseguire mentre il programma sta avanzando.

Breakpoint (Punto di arresto)
Locazione alla quale il programma viene fermato così da poter esaminare i risultati parziali.

Buffer
Area di memoria usata temporaneamente per contenere una informazione che deve essere trasferita tra due dispositivi o tra un dispositivo e la memoria. Un buffer è spesso un registro speciale o una data zona di memoria.

Bug (Imperfezione)
Istruzione o sequenza di istruzioni di un programma che causa indesiderati ed inaspettati risultati.

Bus
Uno o più conduttori utilizzati per la trasmissione di segnali o dell'alimentazione

da una o più sorgenti ad una o più destinazioni.

Byte
Insieme di otto bit considerati come una unità, di solito rappresentanti un carattere alfanumerico.

Call (Chiamata)
Trasferimento del controllo ad una specifica routine.

Carry (Riporto)
Riporto di una somma binaria.

Carry Look Ahead
Tipo di sommatore in cui sono esaminati gli ingressi a più stadi e simultaneamente sono generati gli appropriati riporti.

Central Processor Unit (Unità Centrale di Processo o CPU)
Parte di un sistema di calcolo contenente le unità logiche e aritmetiche, unità di controllo, generatori di sincronismo, memoria ed interfacce di I/0.

Character (Carattere)
Singola lettera, numero o simbolo, usato per rappresentare l'informazione.

Check Bit (Bit di controllo)
Digit binario di controllo, ad esempio il bit di segno.

Chip
Piccolo circuito integrato, racchiuso in un contenitore, composto da molti elementi logici.

Clear (Azzerare)
Cancellare il contenuto di una locazione di memoria mediante la scrittura in essa di tutti zero.

Clock
Segnale avente una frequenza fissa, utilizzato per la sincronizzazione di tutte le operazioni di un dispositivo o computer.

Code (Codice)
Insieme di simboli e regole usate per rappresentare l'informazione.

Combinational Logic System (Sistema Logico Combinatorio)
Sistema digitale privo di elementi di memoria.

Comparator (Comparatore)
Un circuito che confronta due dati e fornisce una indicazione della loro uguaglianza o disuguaglianza.

Compatibility (Compatibilità)
Completa capacità di interscambio tra due dispositivi per realizzare le stesse funzioni o accettare gli stessi programmi.

Compile (Compilare)
Tradurre un programma sorgente simbolico in un programma codificato.

Compiler (Compilatore)
Programma che traduce il linguaggio sorgente ad alto livello in un linguaggio

comprensibile alla macchina.

Computer
Dispositivo capace di eseguire operazioni predefinite e fornire risultati elaborando le informazioni immagazzinate.

Conditional Jump (Salto condizionato)
Salto che avviene quando sono verificate certe condizioni nell'esecuzione del programma.

Console
Quadro di controllo di un computer.

Control Character (Carattere di controllo)
Carattere che all'occorrenza viene utilizzato per iniziare, modificare o fermare una operazione di controllo.

Converter (Convertitore)
Dispositivo che trasforma dati da una forma ad un'altra.

CRC (Controllo Ciclico Ridondante)
Metodo sofisticato di rilevazione degli errori. Il controllo ciclico ridondante è usato nei protocolli di comunicazione moderni ed in molti dispositivi di immagazzinamento dati.

Cross Assembler
Programma assembler che esegue su un computer l'assemblaggio di programmi per un altro computer.

Crosstalk (Diafonia)
Interferenza che si presenta in un dato canale ma che è originata da un altro canale.

CRT Terminal (Terminale video)
Periferico usato per visualizzare insiemi di caratteri, simboli etc.

Current Loop Interface (Interfaccia ad anello di corrente)
Interfaccia usata per connettere terminali direttamente ai computers. Viene adottata nei lunghi collegamenti o in ambienti elettricamente molto disturbati.

Cycle (Ciclo)
Intervallo di spazio o di tempo in cui un set di eventi o fenomeni viene completato.

DAC
Convertitore digitale analogico.

Data Bus (Bus Dati)
Metodo di I/0 per un sistema dove i dati sono convogliati dentro o fuori il sistema digitale per mezzo di un bus comune, connesso a numerosi sottosistemi.

Data Processing (Elaborazione Dati)
Esecuzione di una sequenza sistematica di operazioni effettuate sui dati.

Data Processor (Elaborazione Dati)
Dispositivo capace di effettuare l'elaborazione dei dati.

Data Register (Registro di dati)
Registro contenente dati.

Debug (Mettere a punto)
Rilevare, localizzare e rimuovere errori da una routine o dal malfunzionamento del computer.

Decimal Adjust (Aggiustamento decimale)
Conversione del risultato di una somma binaria in un numero espresso in codice BCD.

Decimal Digit (Cifra Decimale)
In notazione decimale, uno dei caratteri da 0 a 9.

Decoder (Decodificatore)
Dispositivo che riconosce determinate configurazioni binarie, fornendone una opportuna indicazione.

Digit (Cifra)
Un simbolo che rappresenta uno degli interi non negativi più piccolo della radice. Per esempio, in notazione decimale, un digit è uno dei caratteri da 0 a 9.

Digitize (Digitare)
Usare caratteri numerici per esprimere o rappresentare dati.

Direct Access (Accesso Diretto)
Relativo ad un dispositivo di memoria ad accesso casuale.

DMA (Diretto Accesso in Memoria)
I dati vengono trasferiti, via hardware, tra la memoria ed un dispositivo di I/0 senza passare attraverso la CPU.

Dot Matrix (Matrice a punti)
Matrice di punti che è utilizzata per identificare caratteri alfa-numerici.

Double Precision (Doppia Precisione)
Uso di due parole del computer per rappresentare un numero.

Dump
Copiare i contenuti di tutta o di una parte di memoria interna al sistema in una esterna.

Duplex
Metodo di operazione di un circuito di comunicazione in cui ciascuna parte, costituente il sistema può simultaneamente trasmettere e ricevere.

Dynamic Storage Element (Elemento di memorizzazione dinamica)
Elemento di memoria contenente celle di memoria che debbono essere rinfrescate periodicamente per evitare la perdita delle informazioni.

EBCDIC
Individua un codice di 256 caratteri di otto bit usato nella trasmissione binaria dei dati.

Echo (Eco)
Quando un computer rimanda indietro un carattere appena ricevuto al terminale che l'ha rinviato.

Edge Triggering (Triggerato sul fronte)
Attivazione di un circuito con il fronte di salita o di discesa di un impulso.

Edit
Modificare la forma o il formato di dati su un computer mediante un programma.

Editor
Programma utilizzato per creare, modificare programmi su un computer.

EIA RS2 232 Interface (Interfaccia EIA RS232)
Interfaccia utilizzata per collegare terminali e computer. Essa è di solito usata per collegamenti piuttosto corti.

Electrostatic Storage Element (Elemento di memorizzazione statica)
Elemento di memoria che non ha bisogno di rinfresco per conservare i dati.

Emulate (Emulare)
Emulare un sistema con un altro così che il sistema emulante accetta gli stessi dati, esegue gli stessi programmi, e perviene agli stessi risultati del sistema emulato.

Emulator (Emulatore)
Dispositivo che effettua l'emulazione.

End Around Carry
Riporto generato nel posto del digit più significativo e mandato direttamente nel posto del bit meno significativo.

Entry Point
Punto di una routine o programma a cui può essere passato il controllo.

Erase (Cancellare)
Cancellare l'informazione da un mezzo di memorizzazione.

Erasable Programmable Read Only Memory - EPROM
Memoria a sola lettura ripetutamente cancellabile mediante l'esposizione ai raggi ultravioletti.

Execute (Eseguire)
Fase di un ciclo di calcolo (dopo quello di fetch) durante il quale viene selezionata una parola di controllo od eseguita una istruzione.

Fetch (Prelevare)
Parte di un ciclo di calcolo durante cui la prossima istruzione del programma è estratta dalla memoria.

Field (Campo)
Consiste in un'area specificata usata per una particolare categoria di dati.

File
Un insieme logico di dati trattato come una unità. Esso occupa uno o più blocchi di una memoria di massa, quale ad esempio il floppy disk.

File structure
Metodo di registrazione e di catalogazione di files su una memoria di massa.

Firmware
Contenuto di una ROM.

Flag
Una variabile o un registro usato per memorizzare lo stato di un programma o di un dispositivo.

Floating Point Binary Number
Numero binario espresso in notazione esponenziale costituito da una parte mantissa e da una parte esponente.

Floppy Disk (Disco Flessibile)
Disco flessibile magnetico usato per la memorizzazione dei dati.

Flowchart (Diagramma di flusso)
Una rappresentazione grafica per la definizione, analisi o soluzione di un problema in cui i simboli sono usati per rappresentare operazioni, dati, apparecchiature etc.

Format (Formato)
Struttura dei dati.

Fortran
Linguaggio ad alto livello orientato alle applicazioni scientifiche. Abbreviazione di «Formula Translation».

Full Duplex (FDX)
Trasmissione in cui i dati possono essere inviati simultaneamente in due direzioni.

Gate (Porta)
Componente digitale, a più ingressi, ed una sola uscita, che realizza una determinata funzione logica.

Half Duplex (HDX)
Modo di comunicazione che permette la trasmissione dei dati in una sola direzione per volta.

Hamming Code
Sistema di correzione degli errori usato nella trasmissione dati.

Handshake
Processo per cui un segnale di riconoscimento è inviato indietro per confermare il ricevimento delle informazioni.

Hardcopy
Copia stampata.

Hardware
Qualsiasi apparecchiatura fisica che non sia il programma del computer.

Hazard
Transiente d'uscita di un circuito, che causa un valore indesiderato, che appare durante la transizione da uno stato ad un altro.

Integrated Circuit o IC
Qualsiasi dispositivo contenente un chip.

Immediate Address (Indirizzo Immediato)
Pertinente ad una istruzione in cui una parte dell'indirizzo contiene il valore di un operando più che il suo indirizzo.

Indexed Address (Indirizzo Indicizzato)
Un indirizzo che è modificato dal contenuto di un registro indice prima o durante l'esecuzione di una istruzione.

Index Register (Registro Indice)
Registro del microprocessore usato per determinare l'indirizzo effettivo nel modo di indirizzamento indicizzato.

I/O - Input/Output (Ingresso/Uscita)
Operazione di trasferimento dati tra la CPU e periferici o tra periferici e memoria.

Instruction (Istruzione)

Una espressione che specifica una operazione, e i valori o le locazioni dei suoi operandi.

Instruction Register (Registro Istruzione)
Un registro che immagazzina l'istruzione da eseguire.

Interface (Interfaccia)
Dispositivo hardware che permette di connettere fra loro due o più componenti di un sistema.

Interpreter (Interprete)
Programma che traduce gli statements di un linguaggio ad alto livello in linguaggio macchina, eseguendoli istantaneamente.

Interrupt (Interruzione)
Un segnale, che quando attivato causa un trasferimento del controllo ad una data locazione di memoria.

Interrupt Level (Livello d'interruzione)
La priorità assegnata ad una interruzione rispetto alle altre interruzioni.

Jump (Salto)
Operazione che permette di passare da una parte all'altra di programma.

Jump Conditions (Condizioni di salto)
Condizioni definite in una tabella di transizione che determinano il cambiamento di flip-flops da uno stato all'altro.

Keybord
Tastiera
Label (Etichetta)
Uno o più caratteri usati per identificare uno statement o dei dati nella sequenza del programma di un computer.

Lag
Una misura relativa di tempo di ritardo tra due eventi, stati, o meccanismi.

Language (Linguaggio)
Un set di rappresentazioni, convenzioni e regole utilizzate per comunicare informazioni al computer da parte dell'utente.

Library (Libreria)
Un insieme di programmi, routines o subroutines, per mezzo del quale possono essere risolti molti tipi di problemi o parte di essi.

Line Trasmission (Linea di trasmissione)
Qualsiasi conduttore o sistemi di conduttori usati per trasferire delle informazioni da un dispositivo all'altro.

Linker
Programma capace di collegare fra loro diversi moduli e routines per costituire un nuovo programma completo, pronto per essere eseguito.

Load (Caricare)
Caricare informazioni in un computer dalla sua unità d'ingresso.

Loader (Caricatore)
Programma utilizzato per caricare un programma oggetto.

Location (Locazione)
Una locazione di memoria capace di mantenere memorizzata una informazione.

Logic Shift (Scorrimento logico)
Uno scorrimento che interessa tutte le posizioni di un registro.

Loop (Anello)
Ripetizione di un gruppo di istruzioni fino a quando non si verifica una condizione di fine.

LSI
Relativo a circuiti integrati a larga scala di integrazione.

Machine Language (Linguaggio Macchina)
Linguaggio direttamente riconosciuto dal computer.

Macroassembler
Programma Assembler esteso che usa macroistruzioni.

Macroinstruction (Macroistruzione)
Una istruzione in linguaggio sorgente che equivale ad una sequenza di istruzioni macchina.

Macroprogramming
Programmazione con macroistruzioni.

Mainframe
Unità centrale di processo (CPU).

Main Memory (Memoria Principale)
Il set di locazioni di memoria connesse direttamente con la CPU.

Mask (Maschera)
Una struttura di caratteri che è usata per controllare la ritenzione o l'eliminazione di porzioni di un'altra struttura di caratteri.

Mass Storage (Memoria di massa)
Pertinente ad un dispositivo che può immagazzinare un largo ammontare di dati direttamente accessibili alla CPU.

Master Clock
Sorgente di segnali standard digitali richiesti per operazioni sequenziali del computer, di solito consiste in un generatore di impulsi forniti ad ogni fissato intervallo di tempo.

Matrix (Matrice)
Conformazione rettangolare di elementi. Una tabella può essere considerata una matrice.

Memory (Memoria)
Qualsiasi dispositivo in grado di memorizzare dei dati.

Memory Mapping
Un modo operativo del computer in cui i bit più significativi di un indirizzo virtuale sono sostituiti da un valore mutevole che fornisce la ricollocabilità dei programmi.

Microcomputer
Minicomputer avente come CPU uno o più chip a larga scala d'integrazione.

Microinstruction (Microistruzione)
Istruzione, all'interno della CPU, operante ad un livello più basso della istruzione in linguaggio macchina.

Microprocessor (Microprocessore)

Unità centrale di elaborazione in singolo chip.

Microprogram (Microprogramma)
Programma interno alla CPU che permette l'esecuzione dell'istruzione.

Minicomputer
Computer di costo e dimensioni ridotti.

Mnemonic (Mnemonico)
Un simbolo od un nome facilmente ricordabile usato per rappresentare un dato, una istruzione o un codice.

Modem
Dispositivo modulatore-demodulatore usato nella trasmissione dati.

Monitor
Programma di controllo principale che osserva, controlla, gestisce e fa da supervisore a tutte le operazioni di un computer.

Multiplex
Intercalare o trasmettere simultaneamente due o più messaggi su un canale di segnale.

Multiprocessing
Simultanea esecuzione di due o più programmi mediante due o più processori.

Multiprogramming (Multiprogrammazione)
Un metodo di elaborazione in cui più insiemi di istruzioni sono eseguiti ad uno stesso istante.

Negative Logic (Logica Negativa)
Logica in cui la tensione più negativa è rappresentata dallo stato 1; la tensione meno negativa dallo stato 0.

Nesting
Struttura di programmi in cui una subroutine passa il controllo ad un'altra e così via.

Noise (Rumore)
Disturbi indesiderati, in un sistema di comunicazione, che possono generare errori nella trasmissione.

Nondescructive Redaout (Lettura non distruttiva)
Memoria che conserva le informazioni anche dopo la sua lettura.

Object Code (Codice Oggetto)
Il linguaggio in cui viene tradotto uno statement.

Object Program (Programma Oggetto)
Programma scritto in un linguaggio macchina.

Off-Line
Concernente una struttura o dispositivi non controllati direttamente dalla CPU.

On-Line
Concernente una struttura o dei dispositivi direttamente connessi e controllati dalla CPU.

Operand (Operando)
Il dato su cui opera una istruzione.

Operatyng System (Sistema Operativo)

Software che provvede al controllo, compilazione, caricamento ed esecuzione dei programmi utente e altri compiti.

Operation Code (Codice Operativo)
Un codice che rappresenta specifiche operazioni.

Overflow (Traboccamento)
Condizione che si verifica quando il risultato di una operazione aritmetica è troppo grande per essere contenuto in un registro o in una data locazione di memoria.

Pack
Compattare dati in memoria usando un algoritmo per il loro immagazzinamento e recupero.

Page (Pagina)
Sezione contigua di memoria che può essere indirizzata direttamente da una istruzione.

Parallel Operation (Operazione Parallela)
Tutti i digits di una parola sono trasmessi simultaneamente su linee separate per aumentare la velocità di operazione.

Parity Bit (Bit di parità)
Un bit di controllo per il test di validità dei dati.
Se il numero di 1 in un byte è pari, il bit di parità assume il valore 0 se dispari il valore 1 (parità pari) o viceversa (parità dispari).

Patch
Modificare un programma cambiando il codice binario anziché il codice sorgente.

Peripheral (Periferico)
Ogni dispositivo che lavora in congiunzione con un computer ma che è distinto da esso.

Point-to-Point Connection (Connessione punto-punto)
Una configurazione in cui è stabilita una connessione fra due, e soltanto due, installazioni terminali.

Polling
Operazione sequenziale di test dello stato di più dispositivi effettuata per stabilire quello che richiede un servizio.

Port (Porto)
Dispositivo che separa il bus dei dati della CPU da un periferico. È utilizzato per operazioni di I/O.

Positive Logic (Logica Positiva)
Logica in cui il livello di tensione più positivo rappresenta lo stato 1, il meno positivo lo stato 0.

Power Fail Option (Protezione da caduta di alimentazione)
Possibilità hardware che salva automaticamente i contenuti dei registri della CPU in una memoria non volatile, quando viene a mancare la tensione di alimentazione, e li ripristina quando la tensione viene fornita di nuovo.

Priority (Priorità)
Parametro che determina la priorità di interruzione di un dispositivo rispetto ad altri.

Processor (Processore)

Sinonimo di CPU.

Programmable Logic Array
Un circuito integrato che utilizza matrici ROM per realizzare i termini di somma e prodotto di reti logiche.

Programmable Read Only Memory (PROM)
Memoria a sola lettura programmabile una sola volta dall'operatore.

Program Counter (Contatore di programma)
Registro della CPU contenente l'indirizzo dell'istruzione corrente.

Prom Programmer (Programmatore di Prom)
Dispositivo capace di scrivere le informazioni su di una memoria PROM mediante impulsi elettrici.

Propagation Delay (Ritardo di propagazione)
Ritardo che accumula un segnale nell'attraversare un dispositivo logico.

Protocol (Protocollo)
Un set di regole che governano il formato e la temporizzazione dello scambio di messaggi.

Pseudoinstruction (Pseudoistruzione)
Istruzione in linguaggio assembly che fornisce comandi al programma assembler senza essere tradotta in linguaggio macchina.

PSW-Program Status Word
Insieme di flags che con il contatore di programma definiscono lo stato corrente del microprocessore.

Random Access Memory (RAM)
Memoria ad accesso casuale a lettura/scrittura.

Read Enable (Abilitazione alla lettura)
Segnale di controllo, per un elemento di memorizzazione o una memoria, che attiva l'operazione di lettura.

Read Only Memory (ROM)
Memoria a sola lettura programmata dal costruttore con un programma permanente.

Real Time (Tempo Reale)
Modo di operare di un computer, ad alta velocità, che permette di seguire l'evolversi degli eventi nel mondo reale.

Record
Un set di dati adiacenti trattati come una unità.

Redundancy (Ridondanza)
Tecnica utilizzante più di un circuito dello stesso tipo per realizzare una data funzione.

Refresh (Rinfresco)
Metodo che ripristina la carica persa, per effetto della corrente di fuga, dalla capacità della cella di una memoria dinamica (volatile).

Register (Registro)
Memoria temporanea per dati digitali.

Relative Addressing (Indirizzamento Relativo)

Modo di indirizzamento nel quale l'indirizzo effettivo viene calcolato sommando il contenuto del contatore di programma con un valore di spiazzamento contenuto nell'istruzione.

Relocatable (Ricollocabile)
Programma in cui le istruzioni possono essere inserite ed eseguite in una qualsiasi locazione di memoria.

Reset
Riportare un registro o un dispositivo a zero o ad una specificata condizione iniziale.

Response Time (Tempo di risposta)
Il tempo trascorso tra la generazione dell'ultimo carattere di un messaggio inviato da un terminale e la ricezione del primo carattere di risposta.

Routine (Sottoprogramma)
Un set di istruzioni ordinate in una determinata sequenza tale da portare il computer ad eseguire un desiderato compito.

Sample & Hold Circuit
Un circuito che esegue l'operazione di campionamento di un livello di tensione durante un breve periodo di tempo e lo memorizza accuratamente per un più lungo periodo di tempo.

Scanner
Uno strumento che automaticamente campiona o interroga lo stato di vari processi, condizioni, o stati fisici ed inizia una azione in accordo con l'informazione ottenuta.

Scratch Pad Memory
Una piccola zona di memoria ad accesso molto rapido utilizzata per facilitare la manipolazione di dati.

Segmentation (Segmentazione)
Suddivisione di un programma piuttosto grande in diversi programmi che si richiamano l'un l'altro.

Sequential Access (Accesso Sequenziale)
Un metodo di accesso dati in cui i records o i files sono letti uno dopo l'altro nell'ordine in cui essi appaiono nel file o nella memoria di massa.

Sequential Logic System (Sistema Logico Sequenziale)
Sistema digitale utilizzante elementi di memoria.

Serial Accumulator (Accumulatore Seriale)
Un registro che riceve bit di dati in forma seriale o in sequenza e memorizza temporaneamente il dato per un futuro uso.

Setup Time (Tempo di posizionamento)
Il minimo ammontare di tempo di presenza del dato in un ingresso necessario per la sua accettazione, quando il dispositivo è temporizzato.

Shift (Scorrimento)
Un movimento di dati verso destra o sinistra.

Shift Register (Registro a scorrimento)
Registro in cui il dato memorizzato può essere spostato verso destra o sinistra.

Simplex Mode

Trasmissione possibile in un'unica direzione.

Simulate (Simulare)
Rappresentare il funzionamento di un dispositivo, sistema, o programma di un computer mediante un altro opportuno congegno.

Simulator (Simulatore)
Un dispositivo, sistema o programma di computer che rappresenta determinate caratteristiche del funzionamento di un sistema fisico od astratto.

Single Step (Passo-passo)
Esecuzione di un programma, una istruzione per volta, per mezzo di un controllo esterno.

Skip
Ignorare una o più istruzioni in una sequenza di istruzioni.

Soft Copy
Visualizzazione su terminale video.

Software
Un insieme di programmi procedure e relative documentazioni di un sistema realizzato con computer.

Source Language (Linguaggio sorgente)
Il sistema di simboli e sintassi, facilmente comprensibile dall'utente, che viene usato per descrivere una procedura che il computer può eseguire.

Source Program (Programma Sorgente)
Un programma di computer scritto in linguaggio macchina.

Stack
Zona di memoria in cui i dati sono immagazzinati secondo il metodo LIFO («Last In, First Out») cioè l'ultimo dato scritto è poi il primo ad essere letto.

Statement (Frase)
Una espressione o istruzione in linguaggio sorgente.

Static Storage Elements (Elementi di memorizzazione statici)
Elementi di memorizzazione che perdono le informazioni non appena viene a mancare la tensione di alimentazione.

Stored Program (Programma Memorizzato)
Un insieme di istruzioni memorizzate che specificano l'operazione da eseguire.

Subroutine (Sottoprogramma)
Una routine che può essere parte di un'altra routine.

Synchronous Circuit (Circuito Sincrono)
Un circuito in cui tutte le ordinarie operazioni sono controllate da un temporizzatore (master clock).

System Device
Un termine usualmente riferito ad un dispositivo periferico, quale per es. un terminale o un disk drive.

Task
Un insieme di istruzioni che permettono di eseguire un ben determinato compito.

Teletype (Telescrivente)
Dispositivo comprendente una stampante, una tastiera e se desiderato, un lettore e un perforatore di banda.

Terminal (Terminale)
Un dispositivo di ingresso-uscita (I/O).

Timesharing (Divisione del tempo)
Un metodo di suddivisione del tempo della CPU tra i programmi in esecuzione.

Time Slice
Il periodo di tempo concesso dal sistema operativo per eseguire un particolare programma.

Trace
Una interruzione generata da hardware al verificarsi di una condizione anormale.

Translate (Tradurre)
Trasformare statements da un linguaggio ad un altro senza cambiarne il significato.

Two's Complement (Complemento a due)
Un sistema di rappresentazione dei dati in cui i numeri negativi sono codificati con il complemento a due del loro valore positivo sommato di uno.

USASCII (United States of America Code for Information Interchange)
Codice Standard usato negli Stati Uniti per la trasmissione dei dati.

Utility
Ogni programma di uso generale, incluso nel sistema operativo del sistema, che esegue funzioni comuni.

Variable (Variabile)
Una entità che può assumere ciascuno dei valori di un dato set.

Vectored Interrupt (Interruzione Vettorizzata)
Un modo di interruzione accettato da molti microprocessori.

Volatile Storage (Memorizzazione volatile)
Dispositivo in cui i dati immagazzinati sono persi non appena viene a mancare la tensione di alimentazione.

Word (Parola)
Un insieme di uno o più byte considerati come una entità.

Write Enable (Abilitazione alla scrittura)
Segnale di controllo, per un elemento di memorizzazione o una memoria, che attiva l'operazione di scrittura.

Write Time (Tempo di scrittura)
Il tempo che un appropriato livello deve mantenere sulla linea di abilitazione alla scrittura, mentre il dato è presente, per garantire la successiva scrittura del dato in memoria.

BIBLIOGRAFIA

T. R. BLAKESLEE: *Digital Design with Standard MSI and LSI,* John Wiley & Sons, New York, 1975.
J. D. LENK: *Logic designer's manual, Reston Publishing Company, Reston, 1977.*
J. MILLMAN, H. TAUB: *Pulse, digital and switching waveforms, Mc Graw-Hill, New York, 1965*
CONTE, GIORDANA, DEL CORSO, POZZOLO: *Sistemi a microprocessore,* Boringhieri, Torino, 1979.
AA. VV.: *Metodi di interfacciamento,* Edelektron, Milano, 1981
A. VERONIS: *Microprocessors: Design & Applications,* Reston Publishing Company, Reston, 1978.
OSBORNE, KANE, RECTOR, JACOBSON: *Z 80 Programming for logic design,* Osborne Incompany, Berkeley, 1978.

MANUALI:
Z 80 CPU, PI0 & CTC manuale tecnico, SGS-ATES, Agrate Brianza, 1980.
Z 80 CPU, Set di istruzioni, SGS-ATES, Agrate Brianza, 1980.
Terminals and Communication handbook, Digital Equipment Corporation, Maynard (USA), 1981.
Software handbook, Digital Equipment Corporation, Maynard, 1980.
Microcomputer interfaces handbook, Digital Equipment Corporation, Maynard 1981.
Microcomputers and memories, Digital Equipment Corporation, Maynard 1981.
The Interface Circuits Data Book, Texas Instruments, Dallas, 1982.
The TTL Data Book, Texas Instruments, Dallas 1982.
The Optoelectronics Data Book, Texas Instruments, Dallas, 1982.
The MOS Memory Data Book, Texas Instruments, Dallas, 1982.
Data and Design Manual, Teledyne Semiconductor, Mountain View, 1981.
Component Data Catalog. Intel Corporation, Santa Clara, 1982.
The European CMOS Selection, Motorola Inc., Austin, 1979.
Semiconductor Data Library, Motorola Inc., Austin (Texas), 1975.
m2900 Bipolar (TTL) Processor Family, Motorola Inc., Austin, 1976.
COS/MOS Integrated Circuits, RCA Corporation, Sommerville (USA), 1980.
The IC Professionals Signetics Corp., Sunnyvale (Cal), 1973.
Interface Databook, National Semiconductor, Santa Clara, 1979.
Corso sui microprocessori, Rad - Tronic, Torino 1980.

RIVISTE
Linea EDP Alfa Linea Milano, 1982.
Elettronica Oggi, Gruppo Editoriale Jackson, Milano, annate 1980, 81 ed 82.
Electronic Design, L. Altman Publisher, Rochelle Park (N J), annate 1980, 81, 82.

M

Questo volume, sprovvisto del talloncino a fronte, è da considerarsi copia di **saggio-campione gratuito,** fuori commercio (D.P.R. 26 ottobre 1972, n. 633, art. 2. 3° comma, lett. d). Esente da bolla di accompagnamento (D.P.R. 6 ottobre 1978, N. 627, art. 4, n. 6).

L. 20.000

MARCO COPPELLI
BRUNO STORTONI

MICROPROCESSORI TEORIA E PROGETTO